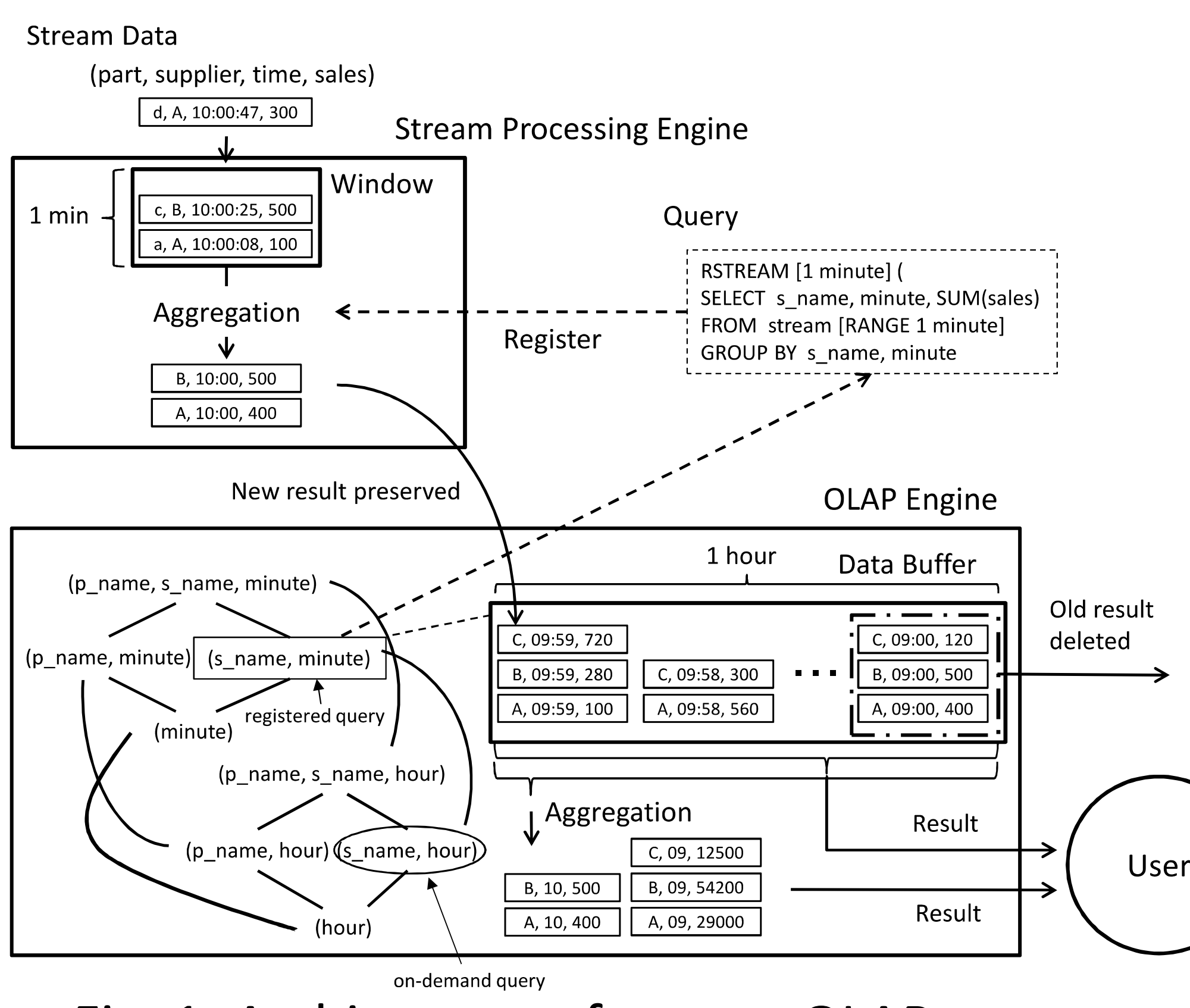
Database Group

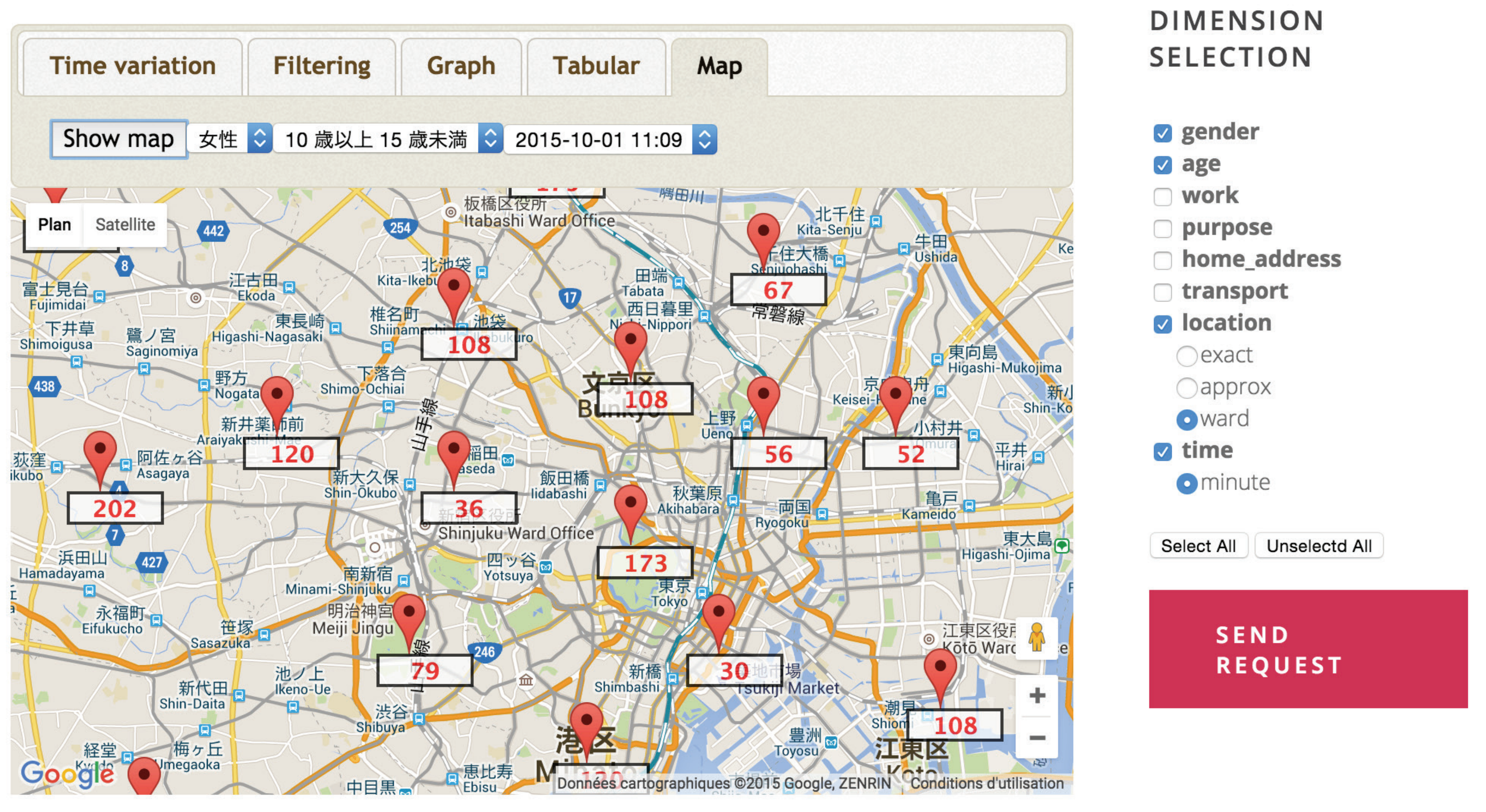
Architecture for Stream OLAP Exploiting SPE and OLAP Engine

Explosive increase of real-time data sources, so-called "data streams" (or just "steams") and increasing demands for real-time analysis over streams give rise to real-time analysis over streams. However, developing tailor-made systems for such applications is not always desirable due to high developing costs and long developing periods. To cope with this problem, we propose a novel architecture for online analytical processing (OLAP) over streams exploiting off-the-shelf stream processing engine (SPE) combined with OLAP engine. It allows users to perform OLAP analysis over streams for the latest time period, called Interval of Interest (IoI). The system in the meantime processes multiple continuous query language (CQL) queries corresponding to different aggregation levels in cube lattice. To cover arbitrary aggregation levels using limited system's memory, we propose to partially deploy CQL queries for those with higher reference frequencies, whereas the results are dynamically calculated using existing aggregation results with the help of OLAP engine. For optimal CQL query deployment, we propose a cost-based optimization method that maximizes the performance. The experimental results show that the proposed architecture is feasible enough to realize stream OLAP by combining an SPE and an OLAP engine. Also, the proposed system significantly outperforms other comparative methods by generating optimized query deployment plans.



```
RSTREAM [1 minute] (
SELECT  s_name, minute, SUM(sales)
FROM  stream [RANGE 1 minute]
GROUP BY  s_name, minute
```

Fig. 1: Architecture of stream OLAP system.



DIMENSION SELECTION

- [✓] gender
- [✓] age
- [ ] work
- [ ] purpose
- [ ] home_address
- [ ] transport
- [✓] location
  - exact
  - approx
  - [✓ ward]
- [✓] time
  - [✓ minute]

Select All | Unselectd All

SEND REQUEST

INTERVAL OF INTEREST

- [✓ All data available]
- The most recent: minutes data
- Fixed time:
  From
  To

Fig. 2: A GUI Screenshot of Stream OLAP system analyzing "People Flow Data."

GPU Acceleration of Set Similarity Joins

We propose a scheme of efficient set similarity joins on Graphics Processing Units (GPUs). Due to the rapid growth and diversification of data, there is an increasing demand for fast execution of set similarity joins in applications that vary from data integration to plagiarism detection. To tackle this problem, our solution takes advantage of the massive parallel processing offered by GPUs. Additionally, we employ MinHash to estimate the similarity between two sets in terms of Jaccard similarity. By exploiting the high parallelism of GPUs and the space efficiency provided by MinHash, we can achieve high performance without renouncing accuracy. Experimental results show that our proposed method is more than two orders of magnitude faster than the serial version of CPU implementation, and 25 times faster than the parallel version of CPU implementation, while generating highly precise query results.

1. Randomly permute rows
2. Save the index of the first *1* for each bin

	R_0	R_1	S_0	S_1	
fast [0]	0	0	0	1	b_0
important [1]	0	1	1	0	
gains [2]	0	1	0	0	
database [3]	1	0	1	0	b_1
are [4]	1	0	1	1	
crucial [5]	1	0	0	0	
gpu [6]	0	1	0	1	b_2
using [7]	0	1	0	0	
transactions [8]	1	0	1	0	

$\longrightarrow$

	b_0	b_1	b_2
R_0	*	3	8
R_1	1	*	6
S_0	1	3	8
S_1	0	4	6

Signature Matrix

$$Sim(X,Y) = \frac{coinciding\_bins}{total\_bins}$$

$Sim(R_0, S_0) = 2/3 = 0.67$ (Real similarity: 0.67)
$Sim(R_0, S_1) = 1/3 = 0.33$ (Real similarity: 0.17)
$Sim(R_1, S_0) = 1/3 = 0.33$ (Real similarity: 0.14)
$Sim(R_1, S_1) = 1/3 = 0.33$ (Real similarity: 0.17)

Fig. 3: Similarity estimation using MinHash.

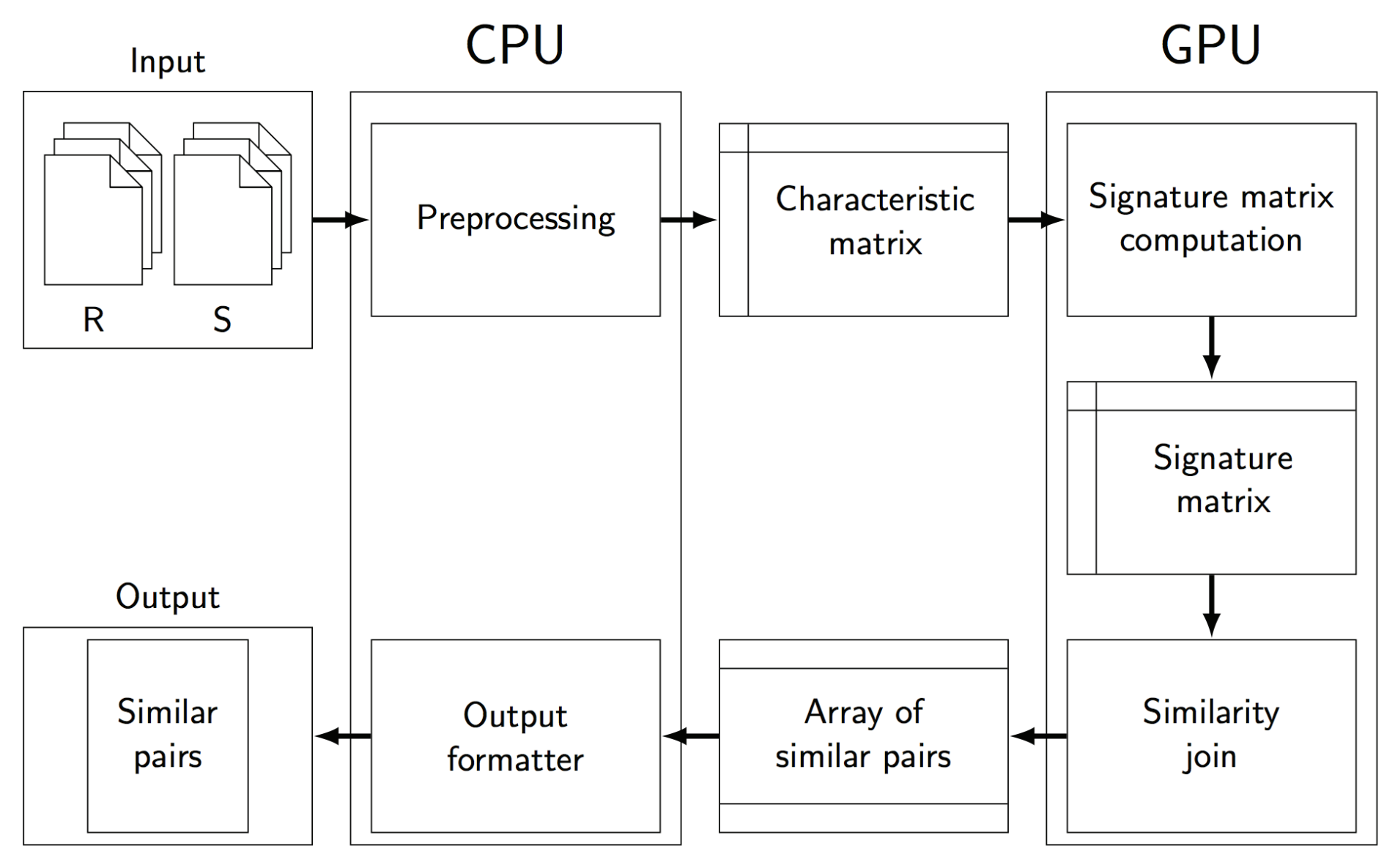
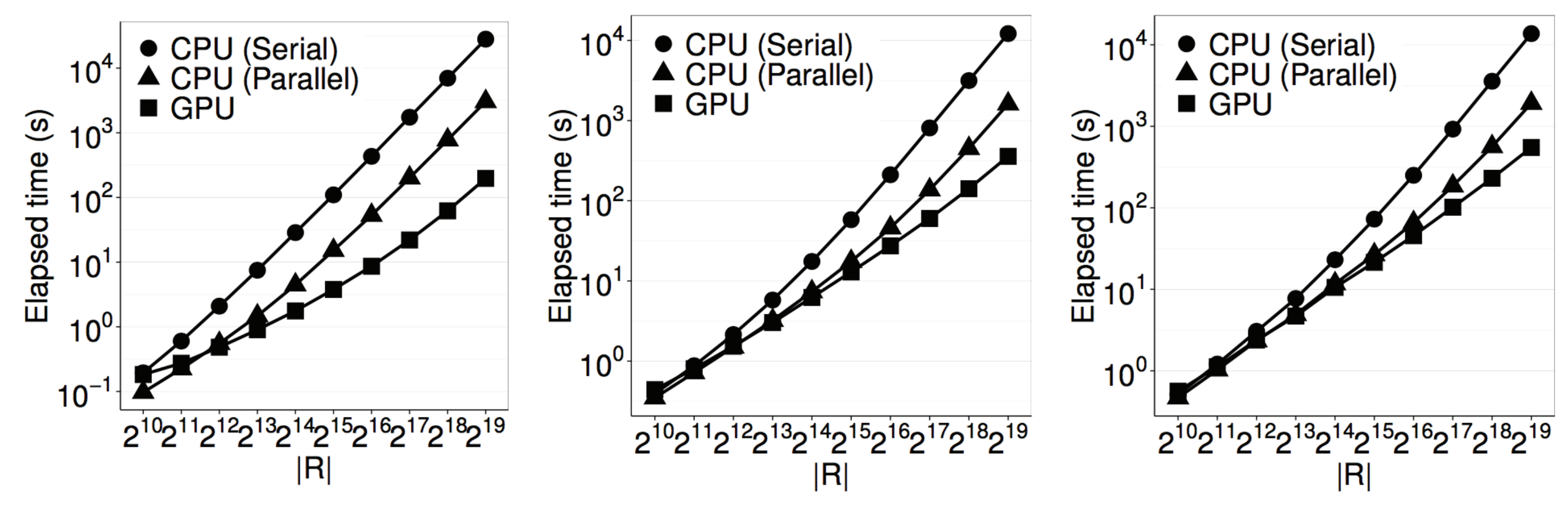


Fig. 4: Process flow.



(a) Images (b) Abstracts (c) Transactions

Fig. 5: Overall performance ($|R| = |S|$)