BotGraph: Large Scale Spamming Botnet Detection

Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum

Presented By: Cole Sherer

Problem

- Web Account Abuse Attack
- Affects Free WebMail Providers:
  - Google
  - AOL
  - HotMail
  - Yahoo!
- Send Billions of Spam Messages

Existing Solutions

- Mail Server Reputation
- Heavy Sender Detection

BotGraph

- Distributed Application
- Input: Large User-User Graph
- Locates Tightly Connected Subgraphs

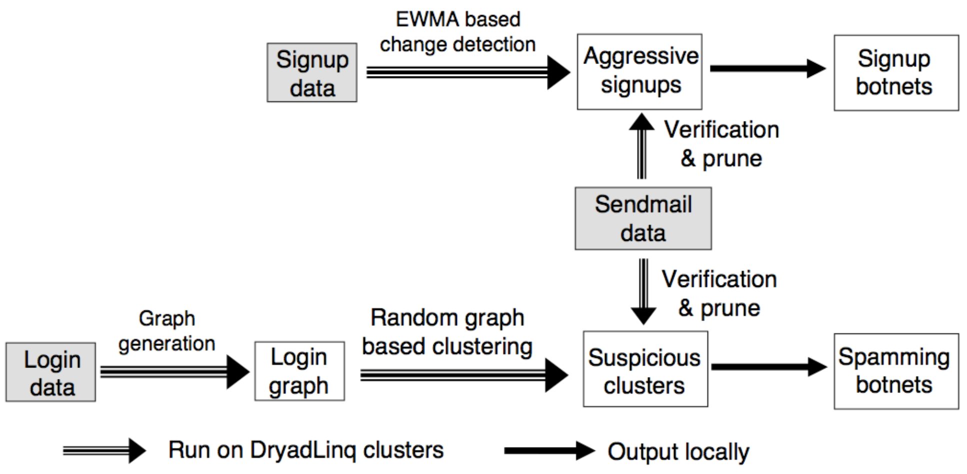
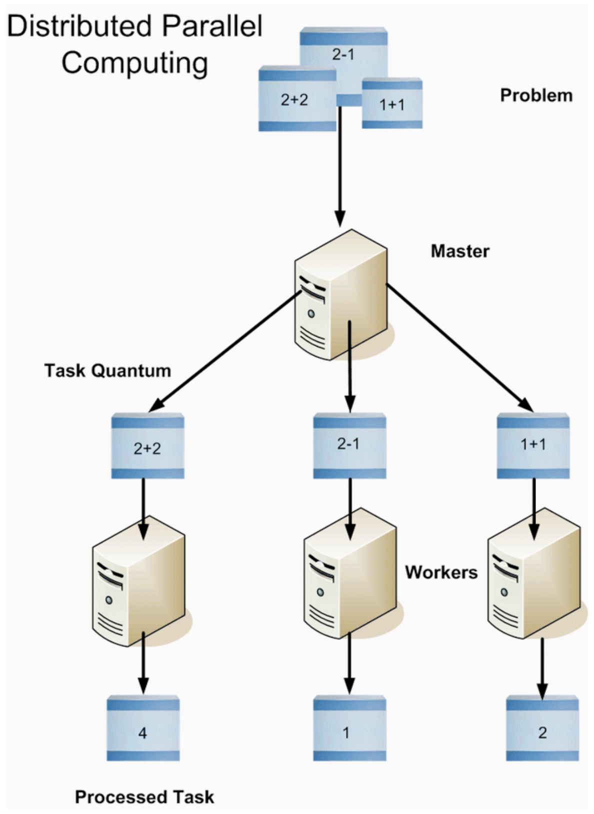


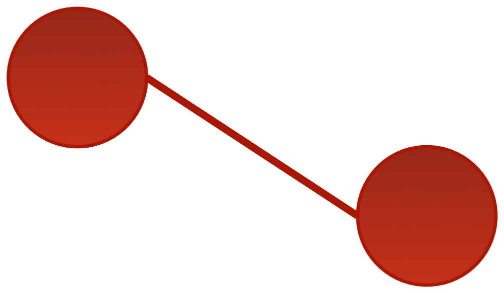
Figure 1: The Architecture of BotGraph.

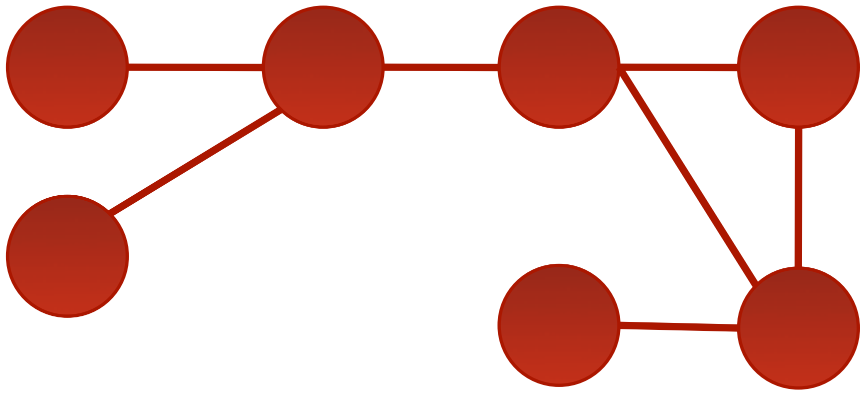
Major Contributions



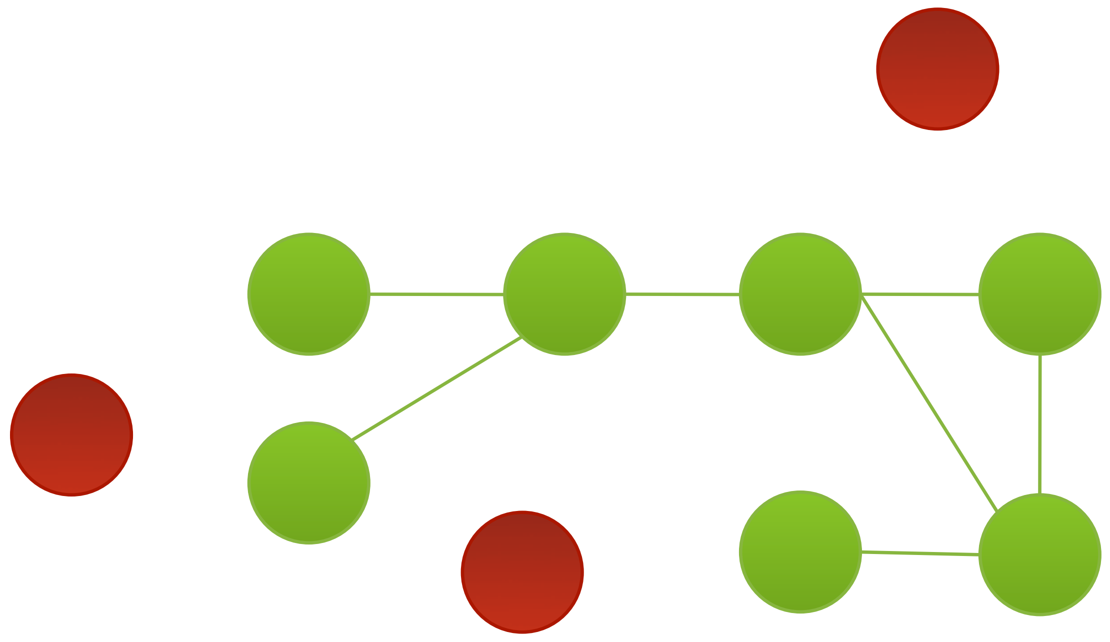
- Novel Graph-Based Detection
- Efficient Implementation with Distributed Programming

Graph Theory Summary

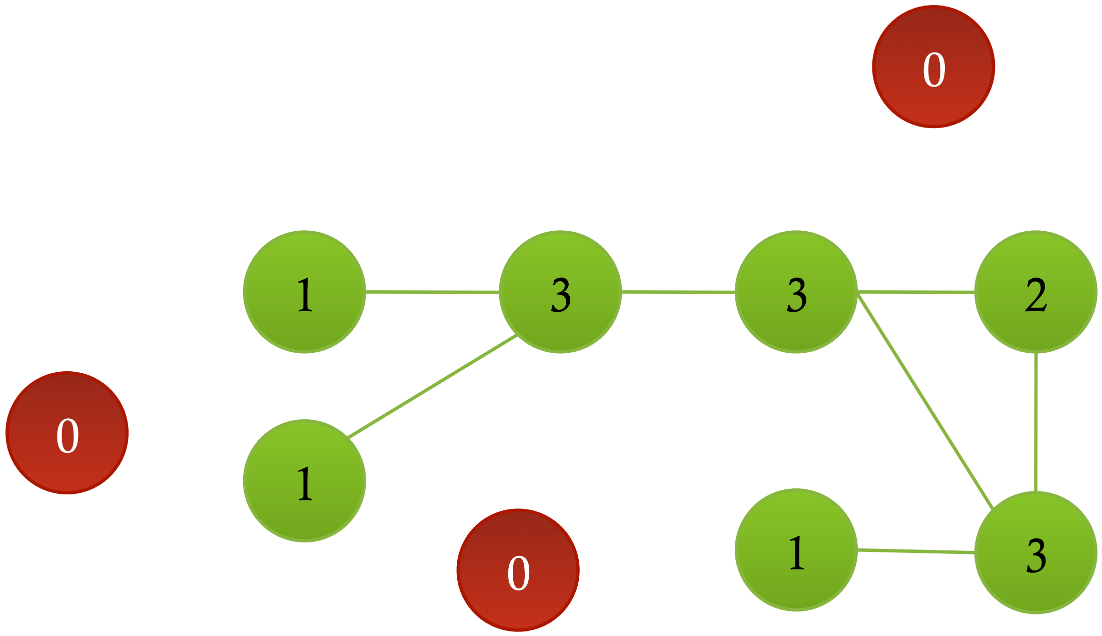

Connected Components


Connected Graph

Giant Component



Degree



Graph Average: 1.4

Random Graph Theory

A graph generated by G(n,p) has average degree d=n*p. If d<1, then with high probability the largest component in the graph has size less than O(log n).

If d>1, with high probability the graph will contain a giant component with size at the order of O(n).

User-User Graph

- Nodes are User Logins
- Edges are Shared IPs
- Edge Weight is Number of Shared IPs

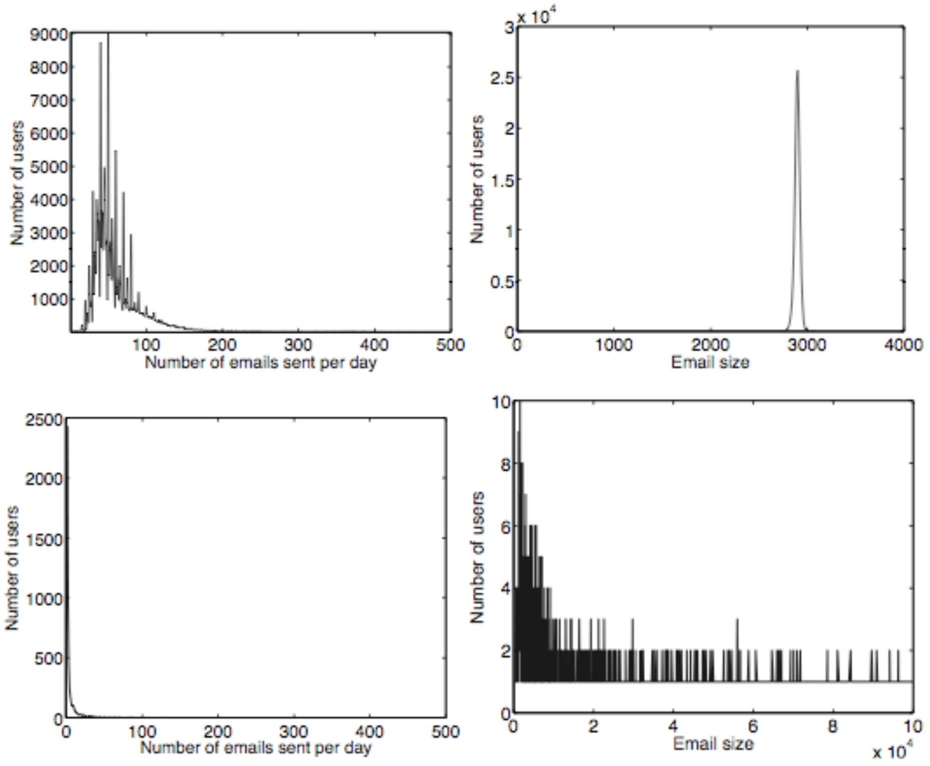
Bot-User Group Tree

```
**procedure** Group_Extracting($G, T$)
1 Remove all the edges with weight $w < T$ from $G$
  and suppose we get $G'$;
2 Find out all the connected subgraphs $G_1, G_2, \cdots,$
  $G_k$ in $G'$;
3 for $i = 1 : k$ do
4     Let $|G_k|$ be the number of nodes in $G_k$;
5     if $|G_k| > M$ then
6         Output $G_k$ as a child node of $G$ ;
7         Group_Extracting($G_k, T + 1$) ;
      end
  end
```

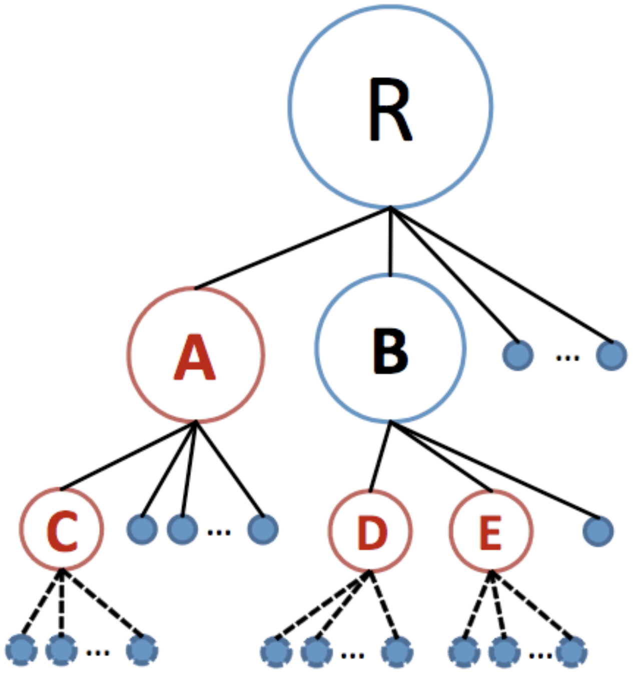
Algorithm 1: A Hierarchical algorithm for connected component extraction from a user-user graph.

Pruning

Bot-Users



Normal Users



Graph Construction Implementation #1

1 Inputs: partitioned data according to IP addresses

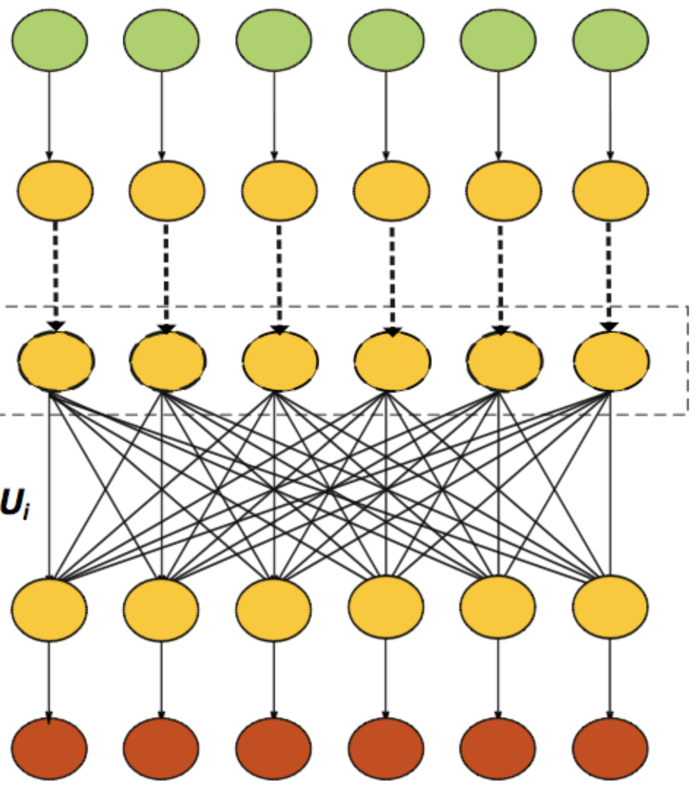
2 For any two users U_i and U_j sharing the same IP, output an edge with weight one (U_i, U_j , *1*)

3 Optional local aggregation step

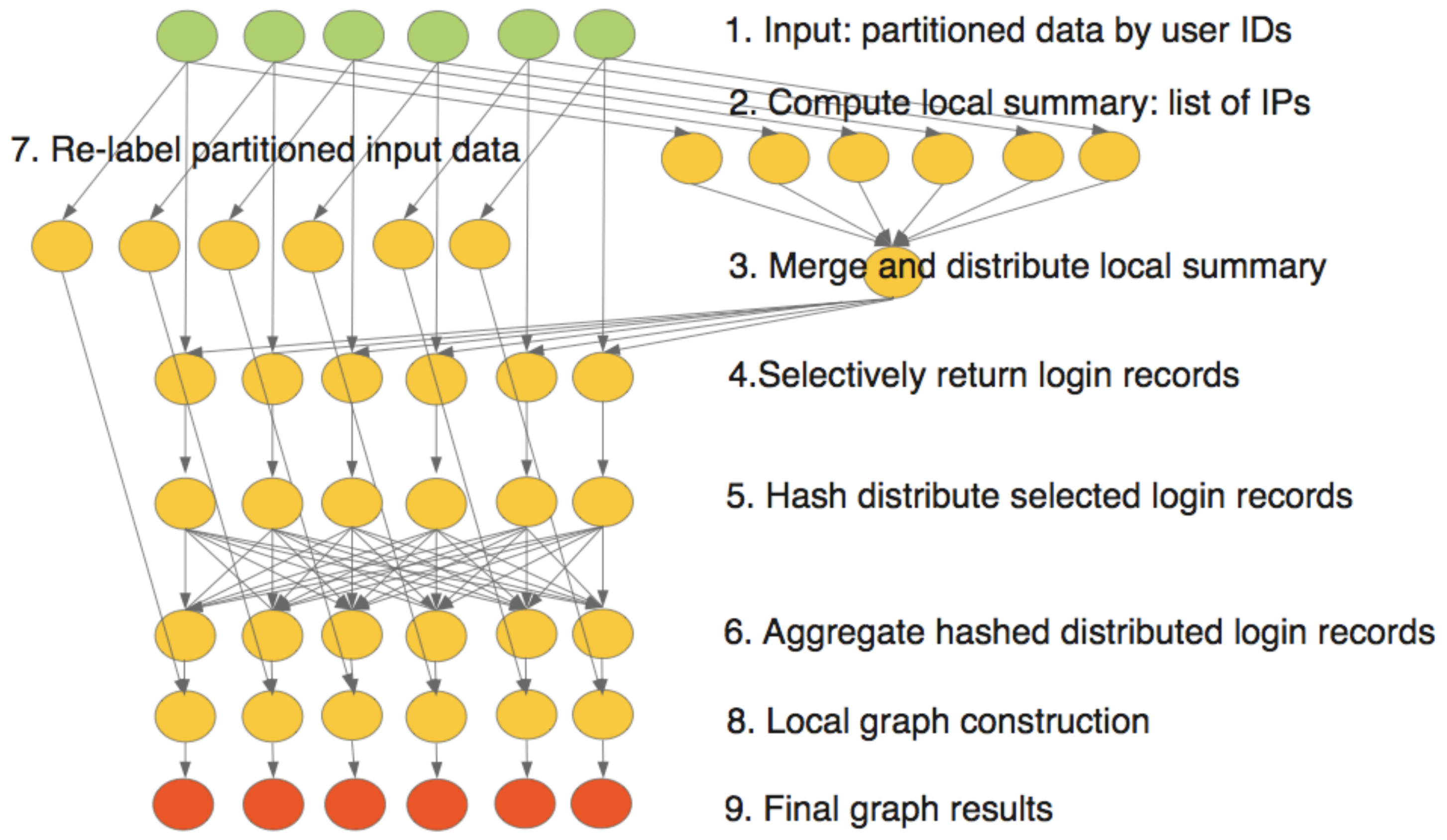
Hash distribute edges according to U_i

4 Aggregate edge weights

5 Final graph results



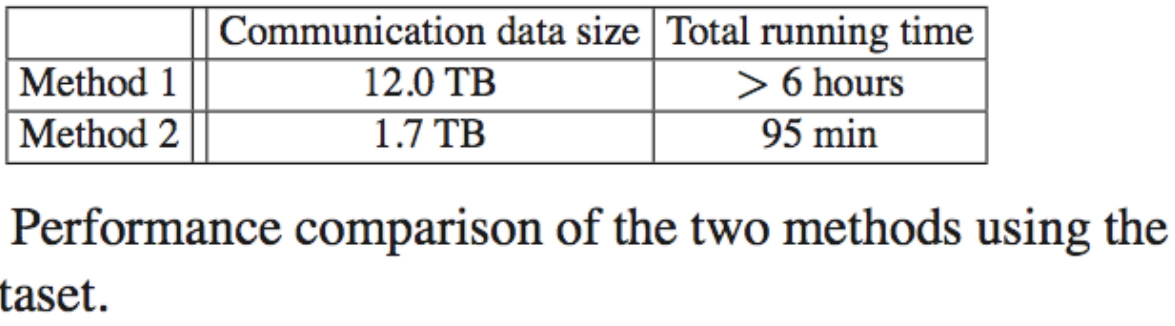
Graph Construction Implementation #2


1. Input: partitioned data by user IDs
2. Compute local summary: list of IPs
7. Re-label partitioned input data
3. Merge and distribute local summary
4.Selectively return login records
5. Hash distribute selected login records
6. Aggregate hashed distributed login records
8. Local graph construction
9. Final graph results

Optimizations

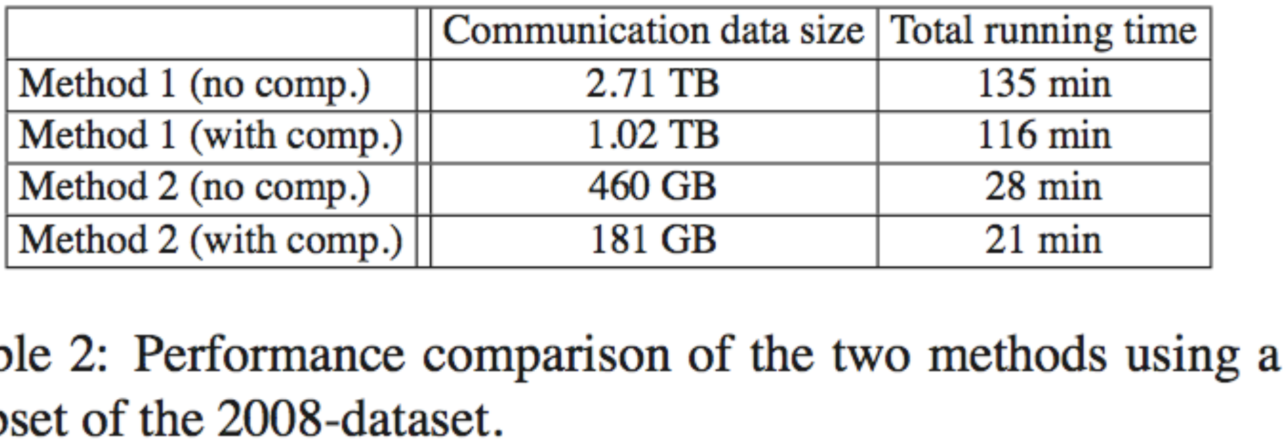
- Pre-Filter Users by Autonomous System
- Compress Communications
- Parallel Data Merge

Results



	Communication data size	Total running time
Method 1	12.0 TB	$>$ 6 hours
Method 2	1.7 TB	95 min

Table 1: Performance comparison of the two methods using the 2008-dataset.



	Communication data size	Total running time
Method 1 (no comp.)	2.71 TB	135 min
Method 1 (with comp.)	1.02 TB	116 min
Method 2 (no comp.)	460 GB	28 min
Method 2 (with comp.)	181 GB	21 min

Table 2: Performance comparison of the two methods using a subset of the 2008-dataset.

Results

- 0.44% False Positive Rate
- Parse a 220GB Hotmail Log in 1.5 hours on 240 Machines (500 Million Nodes – 100s of Billions of Edges)
- Located 26 Million Spam Accounts in 500 Million Total Accounts

Questions