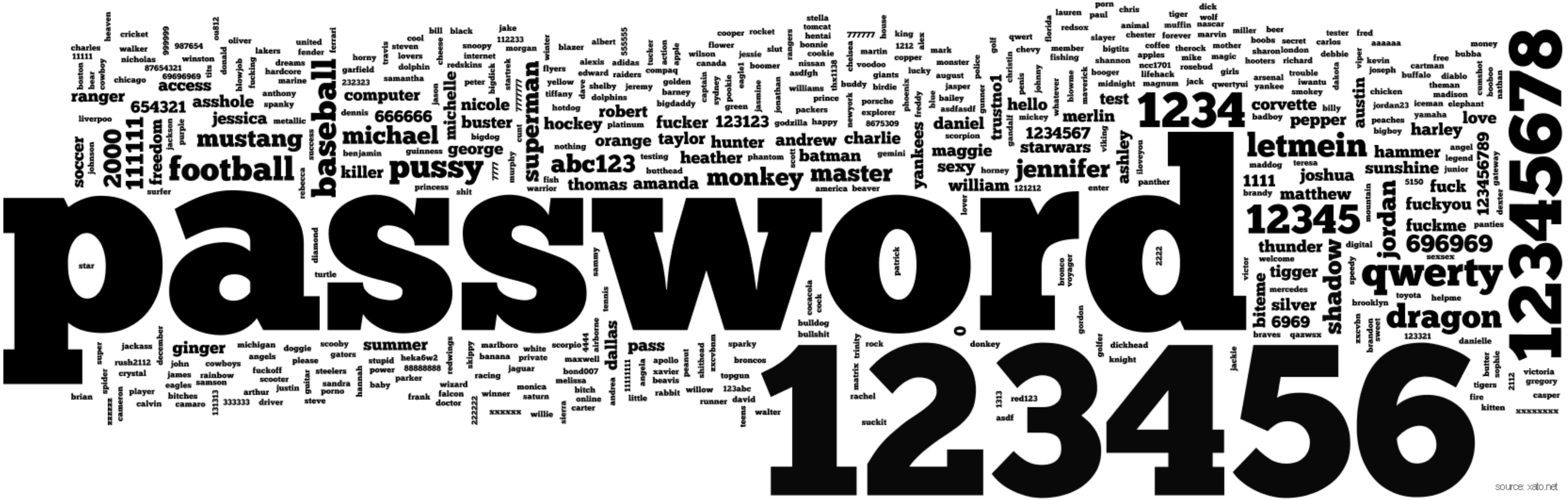
A GRAPH-BASED METHOD FOR CROSS-ENTITY THREAT DETECTION

Herman Kwong, Ping Yan

Salesforce

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Account Takeover



source: xato.net

Detection is Key

- Basic features
- Known signature
- Usage anomaly

Each with their weaknesses

CrossLinks

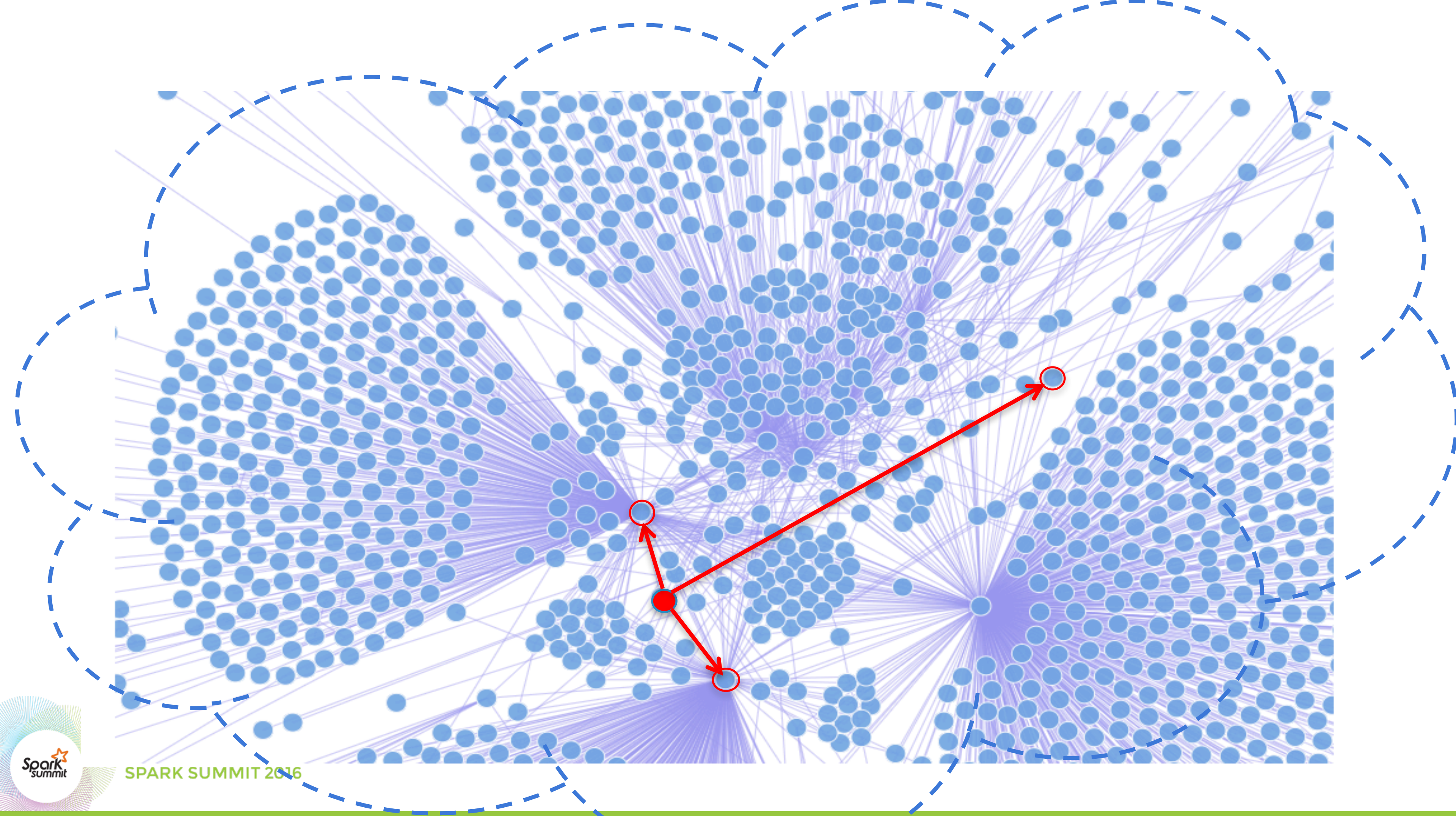
- Unexpected common features, across unrelated user accounts / environments

- Features:

  IP, location, time zone,

  user agent, browser fingerprint,

  user action sequence, ...



Why Graph(X)?

- Why Graph?
  - Classical pair-wise entity relationship measurement solutions require $O(N^2)$ computations
  - Computation complexity dramatically reduced by localizing computations
  - Highly extensible solution with a multigraph

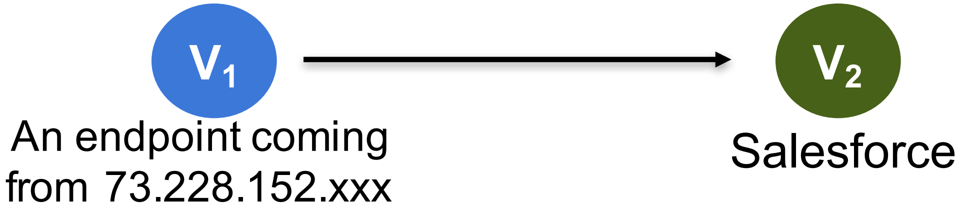
- Why GraphX?
  - Spark ecosystem
  - Scalability and performance
  - Advanced Graph algorithms

Graph-theoretical Techniques

- Graph analysis is of high interest in many social network contexts
  - Proximity-based approaches
  - Personalized pagerank: closeness of each node to the restart nodes
  - Simrank: similarity of contextual structures
- Bridge-Node anomaly [Akoglu, et al 2015]
  - Publication networks: authors from different research communities
  - Financial trading networks: cross-sector traders
  - Customer-product networks: cross-border products
  - Network intrusion detection: cut-vertices indicating nodes accessing multiple communities that they do not belong to

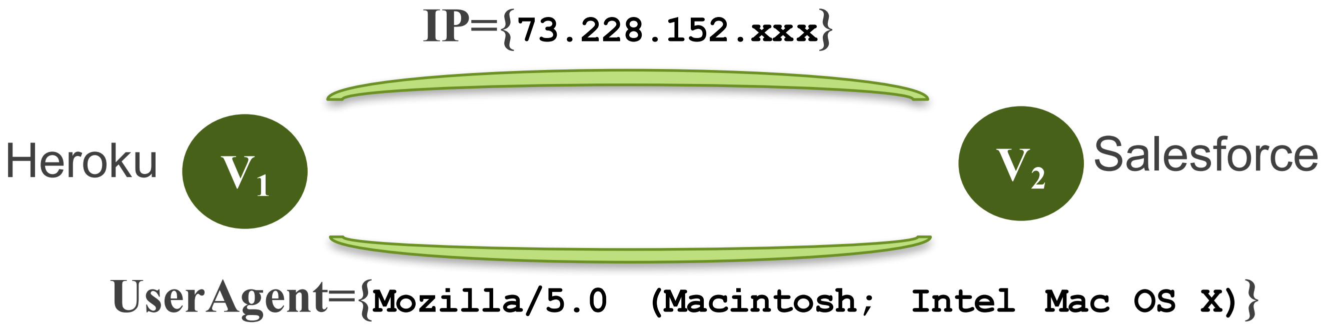
Bipartite Graph

Bipartite: Application access data directly makes a ***bipartite graph*** where an edge represents V_1 accessing V_2



Multigraph Formulation

We can also formulate the relationship of application access data as a ***multigraph*** where an edge between two entities represents some features that the two entities have in common.



Anomaly Detection by Graph Change Detection

- Our objective is to quickly discover changes in the access graph over time

- Unexpected new cross-entity connections are of particular interest in security detection problems

- A naïve detector and a community-based algorithm were proposed for access anomaly detection with a graph

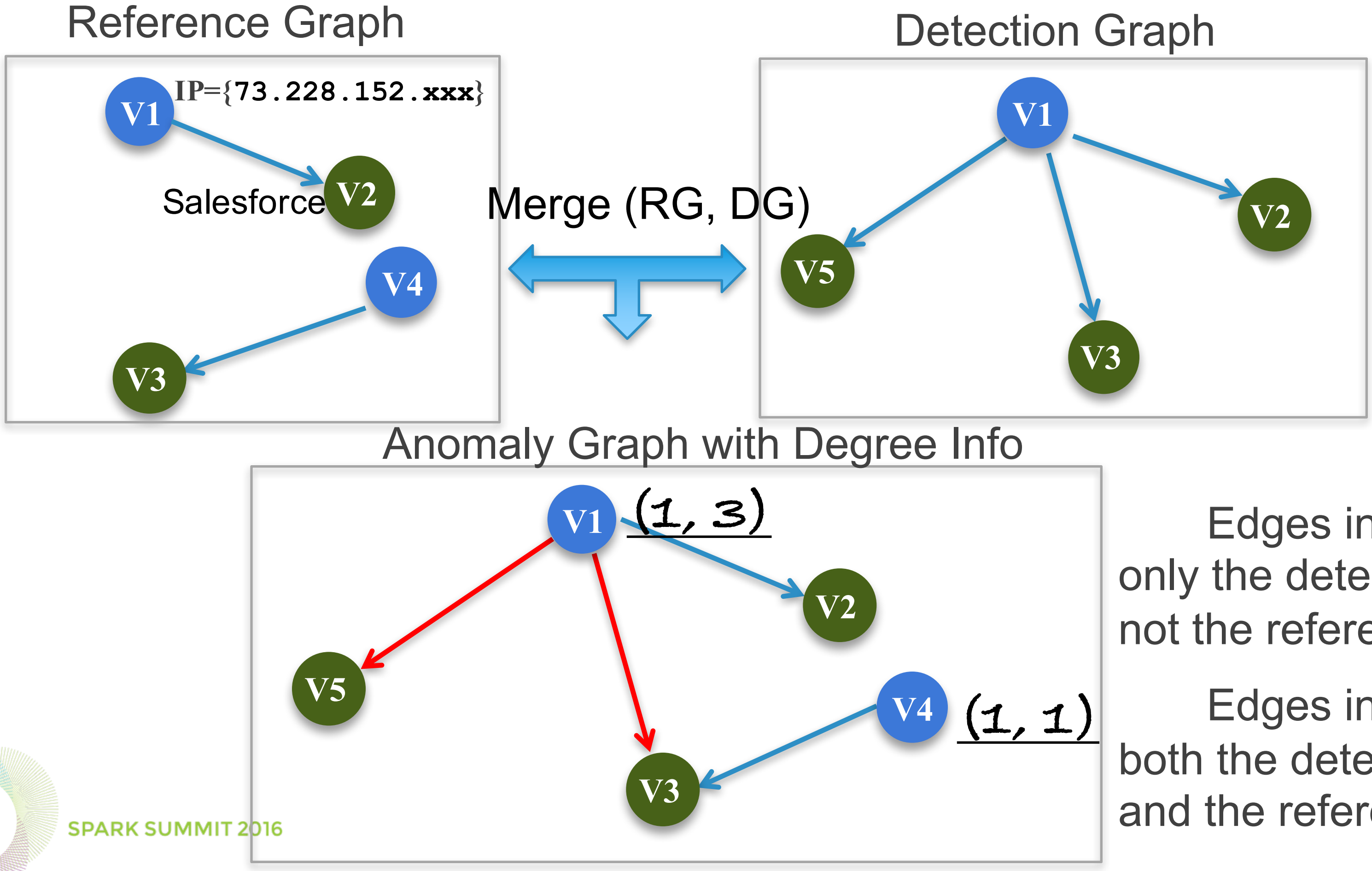
Naïve Detector

REFERENCE GRAPH (TRAINING) - RG
DETECTION GRAPH (TESTING) — DG
MERGE OF RG & DG - RGDG
ANOMALY GRAPH (DEGREE INFO) — AG
CONNECTIVITY GRAPH (ENV-TO-ENV) — CG

Detection:

```
//outDegRG: count of neighbors in test nodes
//outDegRGDG: count of neighbors of nodes in the combination of test
data and reference data
//We like to calculate the difference between the two degree
//properties : outDegRGDG — outDegRG
```

Naïve Detector



Edges in red: edges in only the detection graph but not the reference graph.

Edges in blue: edges in both the detection graph and the reference graph.

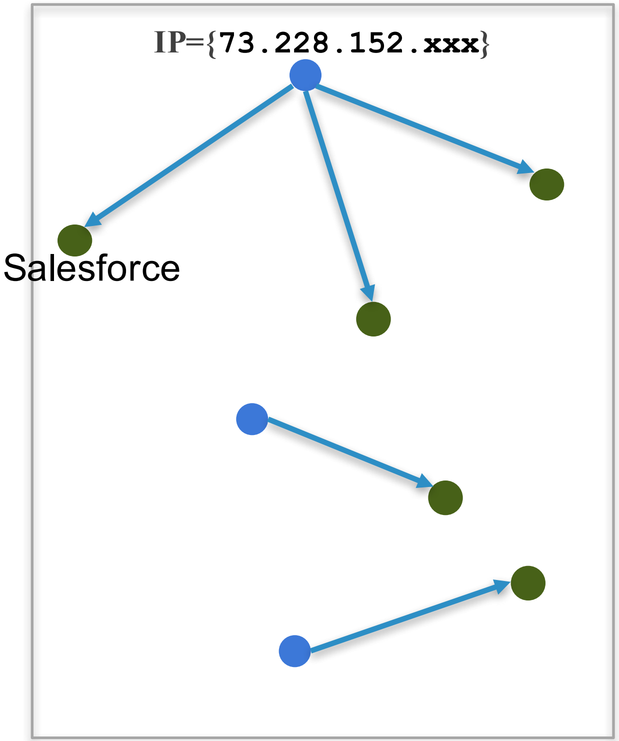
```scala
case class DegreeCnt(feature: String, outDegRG: Int, outDegRGDG: Int)

val initDegreeInfo: Graph[DegreeCnt, Int] =
                    RG.mapVertices{ case (id, feature) => DegreeCnt(feature, 0, 0) }

val DegreeGraph = initDegreeInfo.outerJoinVertices(RG.outDegrees) {
  case (id, u, outDegOpt) => DegreeCnt(u.feature, outDegOpt.getOrElse(0), u.outDegRGDG)
}.outerJoinVertices(RGDG.outDegrees) {
  case (id, u, outDegOpt) => DegreeCnt(u.feature, u.outDegRG, outDegOpt.getOrElse(0))
}
```

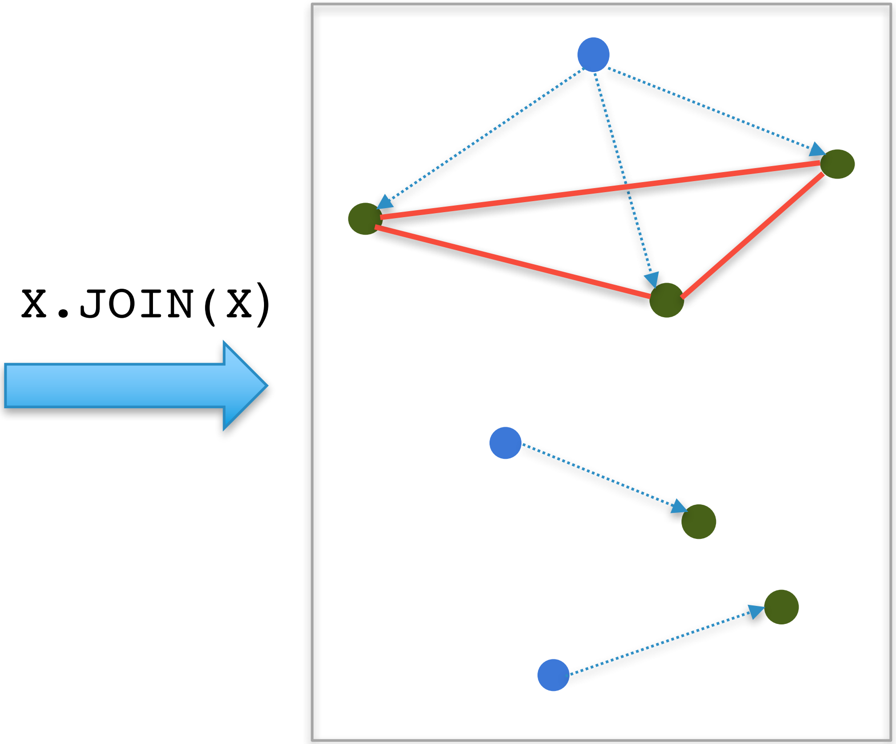
2nd-Order Connectivity

Bipartite graph

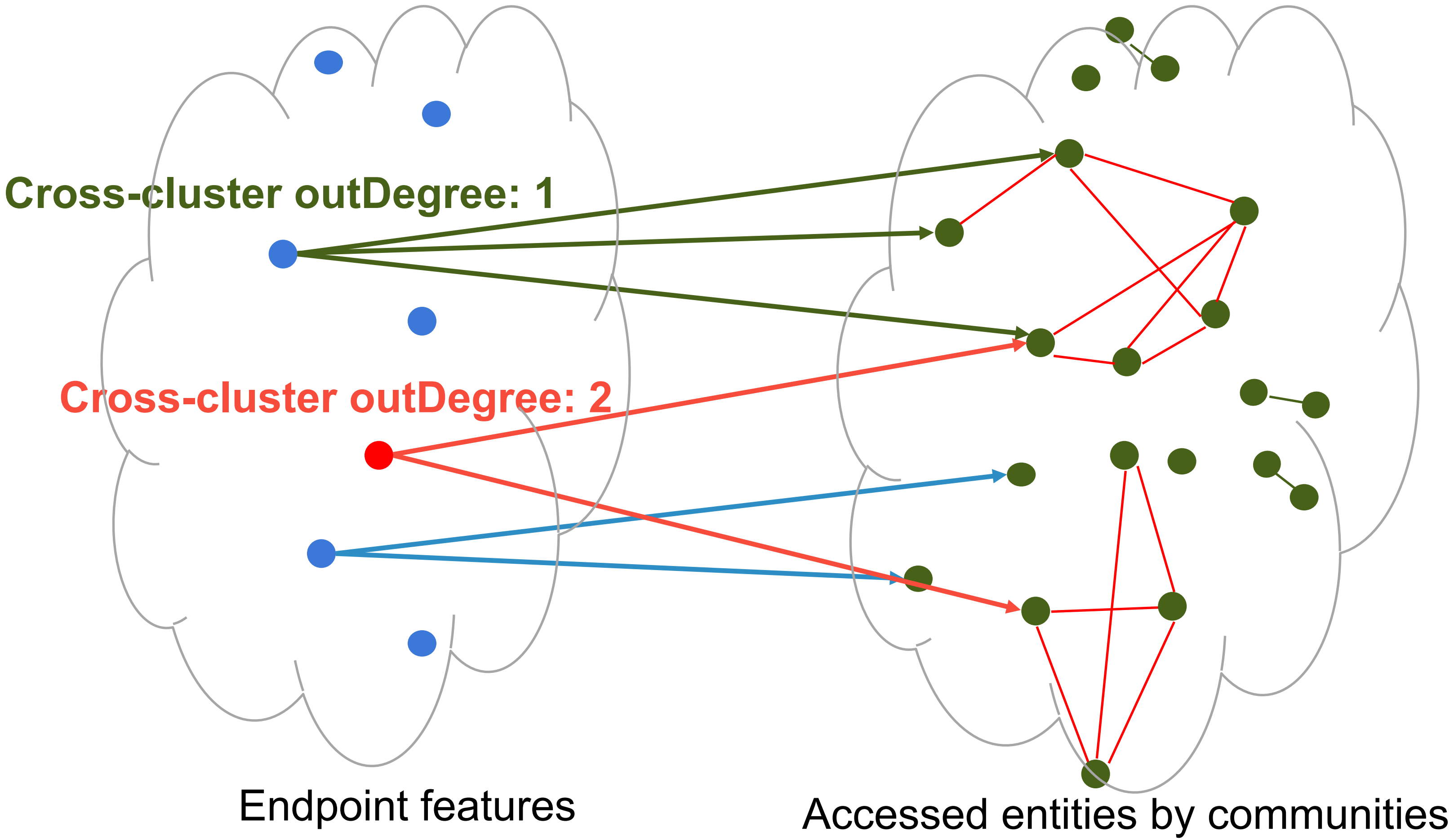


X.JOIN(X)

Connectivity graph



2^{nd}-Order Anomaly Graph



2^{nd}-Order Anomaly Detector

Step 1: self join RG on the *feature-of-interest (e.g., IP)* to get the env-to-env connectivity graph.

Step 2: Build the Anomaly Graph as in the Naïve Detector algorithm (1^{st}-order anomalies).

Step 3*: collapse the cluster of nodes into a single node on the Anomaly Graph.

Step 4: run the naive algorithm to get the updated node degrees to identify 2^{nd}-order anomalies.

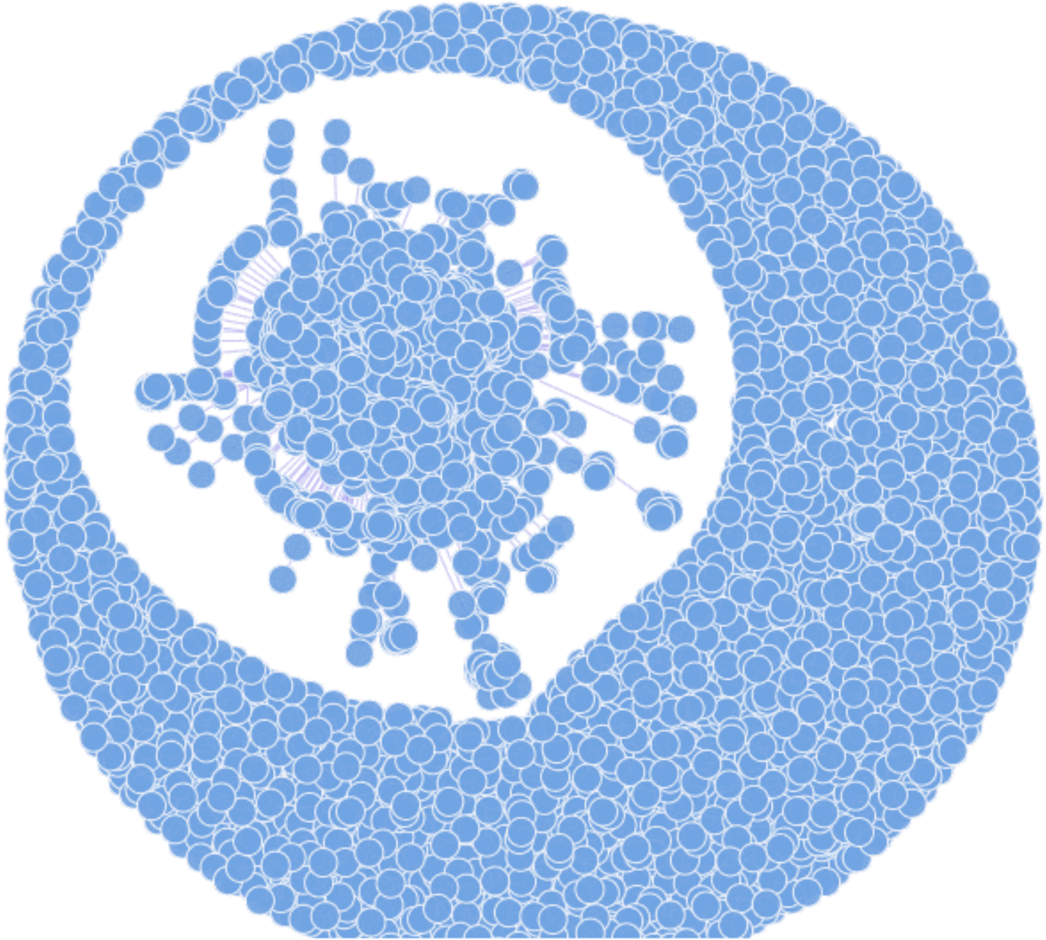
*: ConnectedComponent to approximate clusters

Experiments

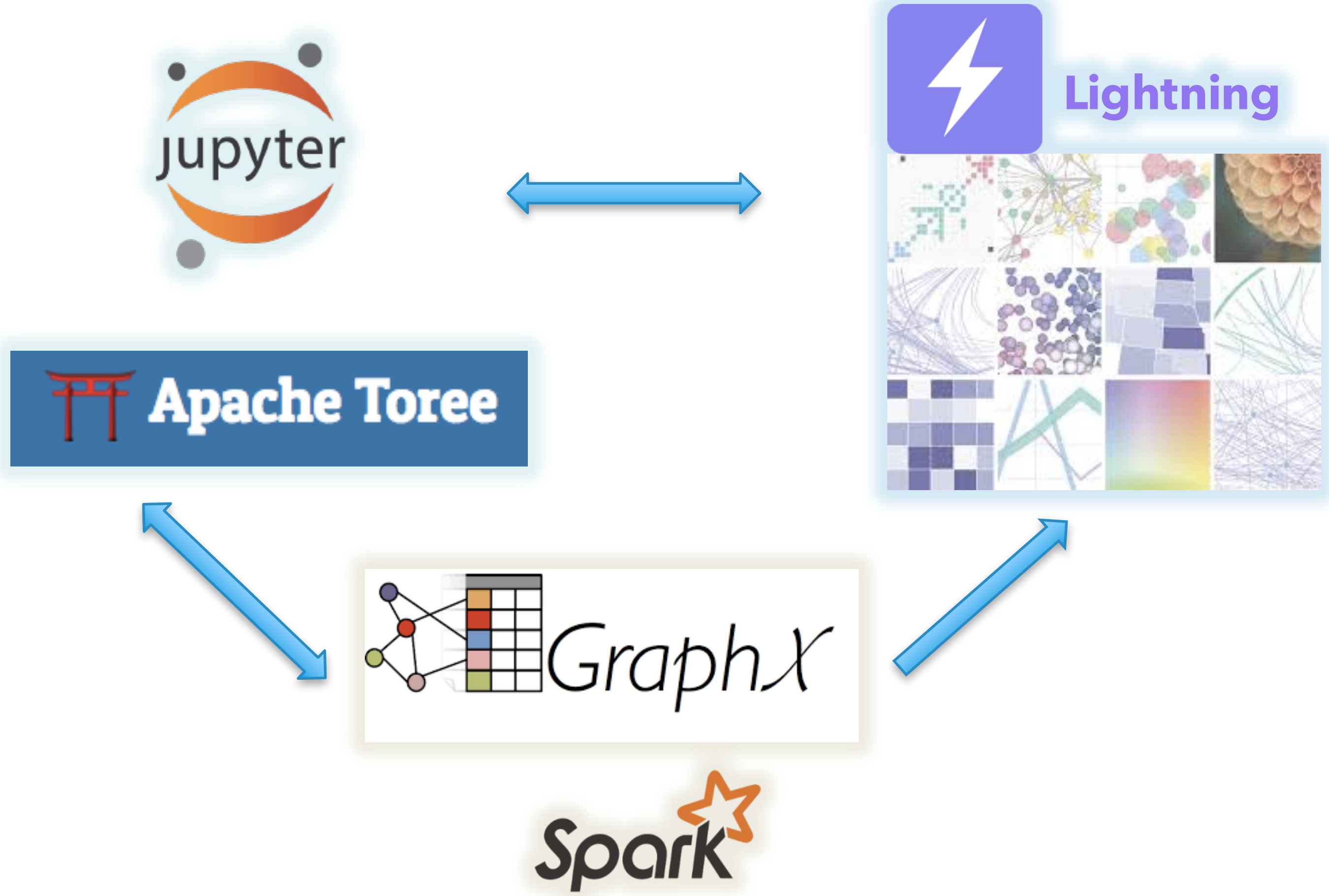
- Reference Graph (RG) - Number of vertices: 2,222,613
- RG - Number of edges 2,156,104
- Connectivity Graph (CG) - Number of vertices: 4,682
- CG - Number of edges: 8,534
- CG - Number of ConnectedComponents: 1146

- Number of 1st-order anomalies: ~700
- Number of 2nd-order anomalies: ~200

- Computing time: ~ 5 minutes on a Mac Air (1.7 GHz Intel Core i7, 8G memory)

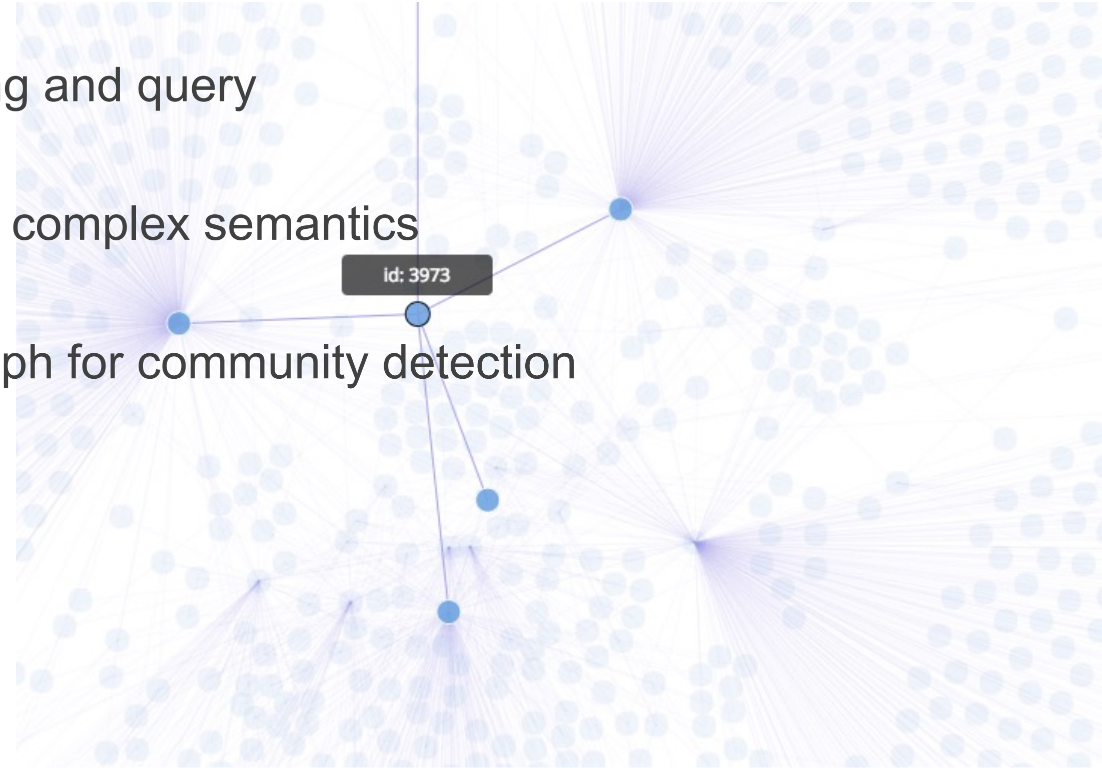


Toolkit for Interactive Analysis



Opportunities

- GraphDB for real-time indexing and query
- Probabilistic edges to support complex semantics
- Clustering on probabilistic graph for community detection



References

[Akoglu et al 2015] Akoglu, Leman, Hanghang Tong, and Danai Koutra. "Graph based anomaly detection and description: a survey." Data Mining and Knowledge Discovery29.3 (2015): 626-688.

[Ding et al 2012] Ding, Qi, et al. "Intrusion as (anti) social communication: characterization and detection." Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012.

THANK YOU.

hkwong@salesforce.com
pyan@salesforce.com (@pingpingya)