Dynamic Community Detection for Large-scale e-Commerce data with Spark Streaming and GraphX

Ming Huang

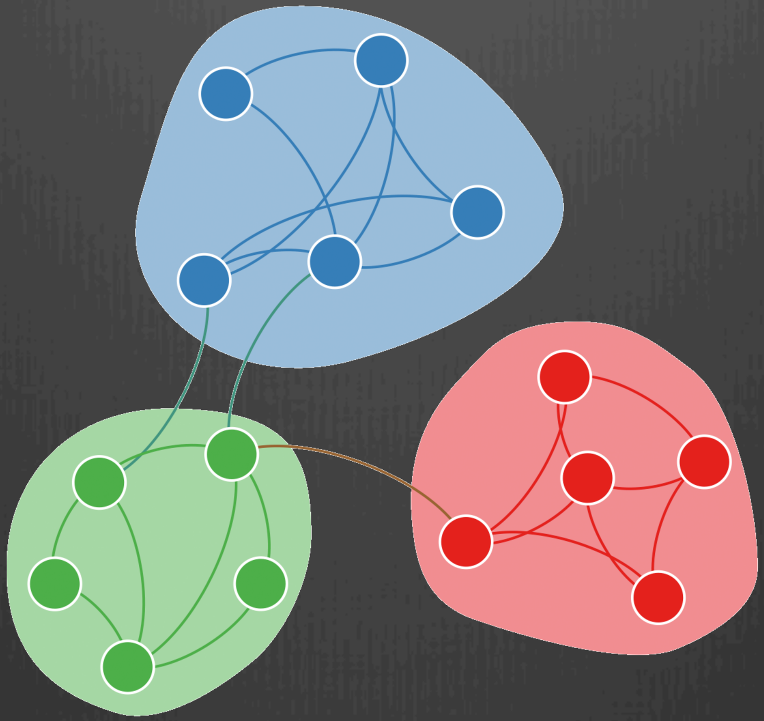
Meng Zhang, Bin Wei

GuangYuan Huang, Jinkui Shi

Community Detection

Scenarios

- VIP Customer
- Reputation Escalator
- Fraud Seller
- ………



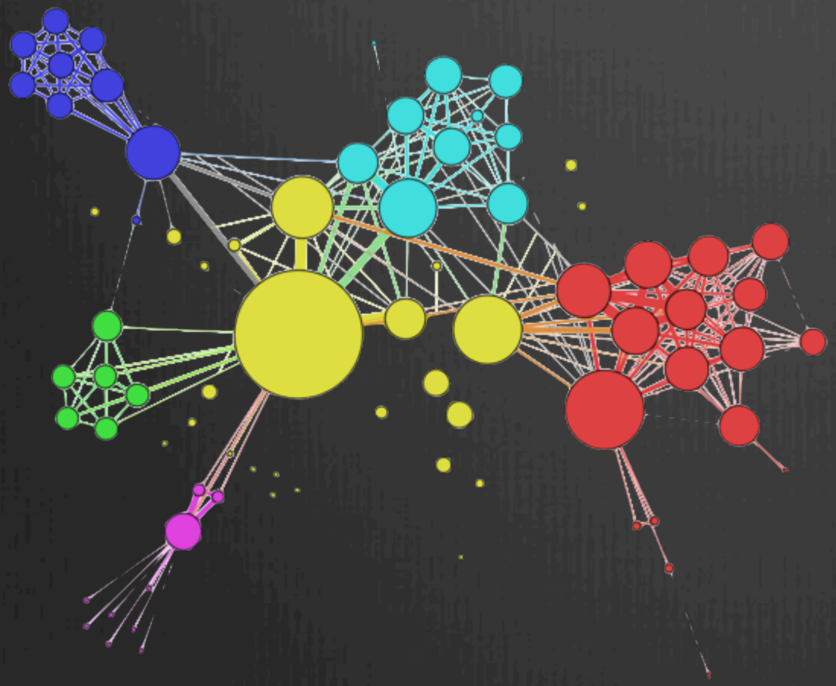
Algorithms

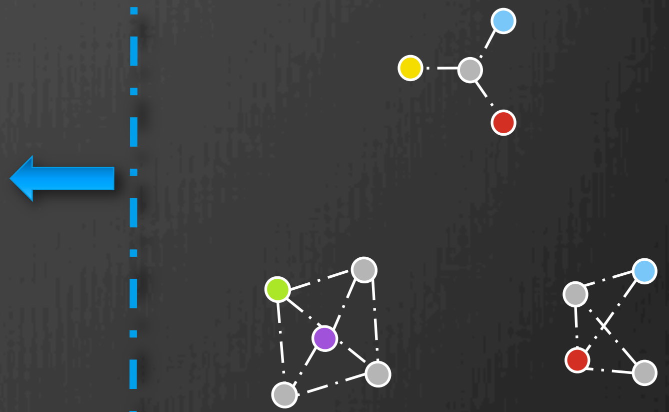
- LPA
- GN
- Fast Unfolding
- …….

How to make it Dynamic?

Static Communities

Streaming Data





Make sophisticated, real-time decisions

Definition & Solution

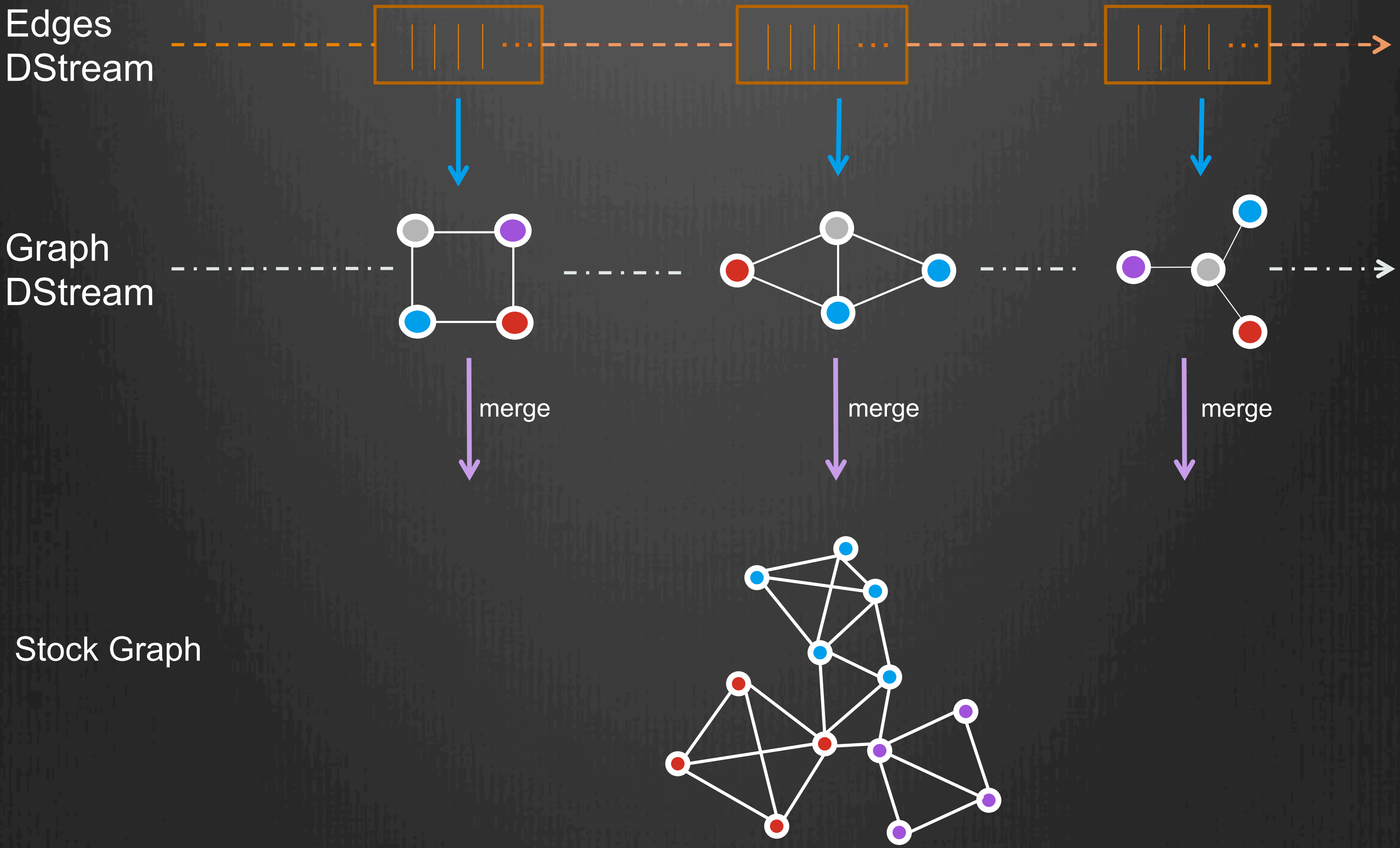
Dynamic Community Detection

1. Decide New Node’s community
2. Update Graph Physical Topology
3. Effect communities and modularity

REAL-TIME

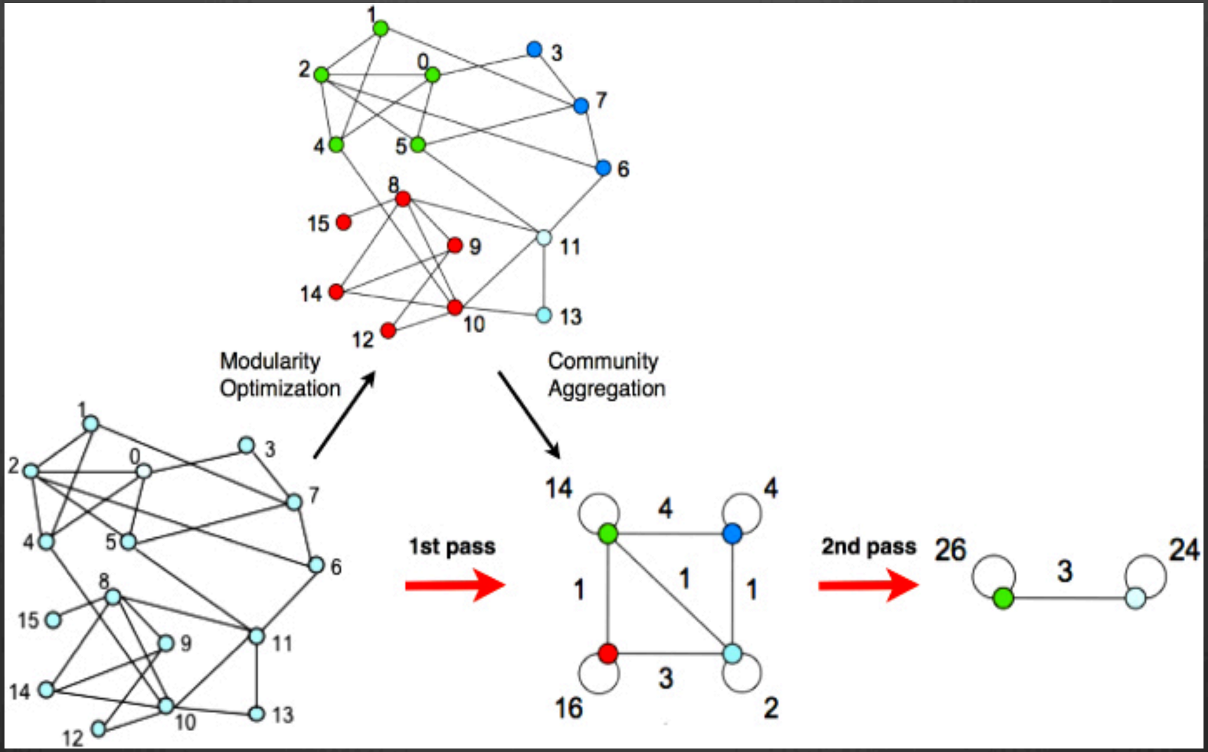
Spark Streaming + GraphX → Streaming Graph

Streaming Graph



Models and Algorithms

Quick Overview of Fast Unfolding



Modularity : $Q = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j)$ $\Rightarrow$ $Q = \sum_{i}^{c} Q_i = \sum_{i}^{c}\left[\frac{\sum in}{2m} - \left(\frac{\sum tot}{2m}\right)^2\right]$

Incremental Algorithms

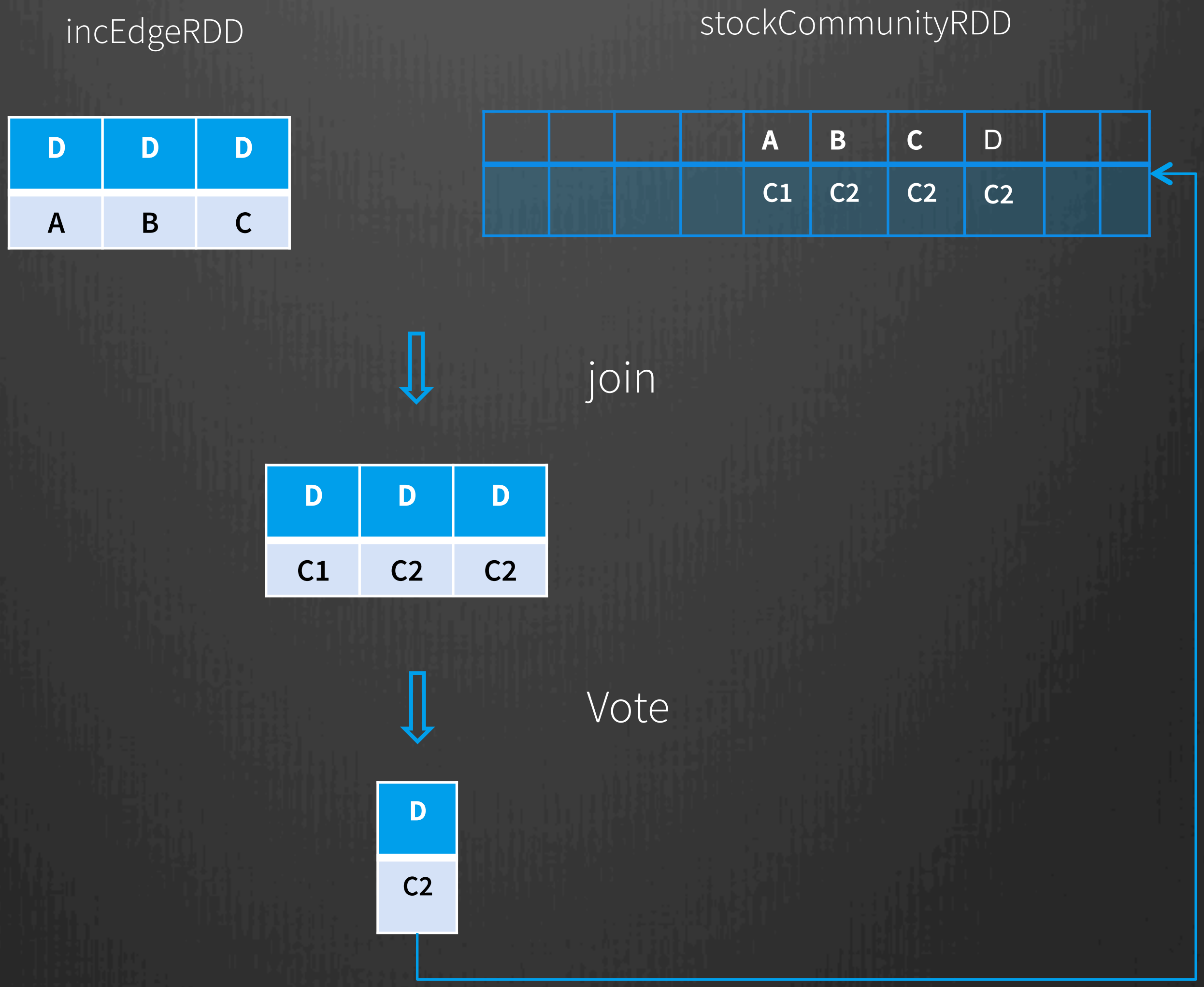
JV（Streaming with RDD）

- Join & Vote

UMG（Streaming with Graph）

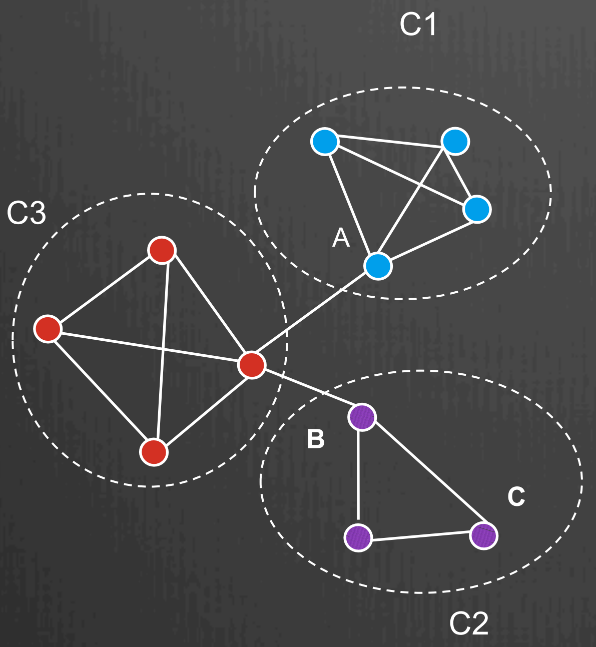
- Union & Modularity Greedy

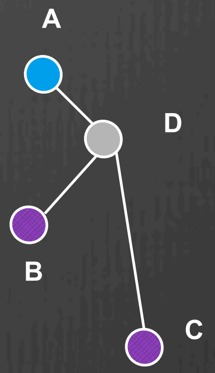
JV



incEdgeRDD
D D D
A B C
stockCommunityRDD
A B C D
C1 C2 C2 C2
join
D D D
C1 C2 C2
Vote
D
C2

UMG 1 - Union

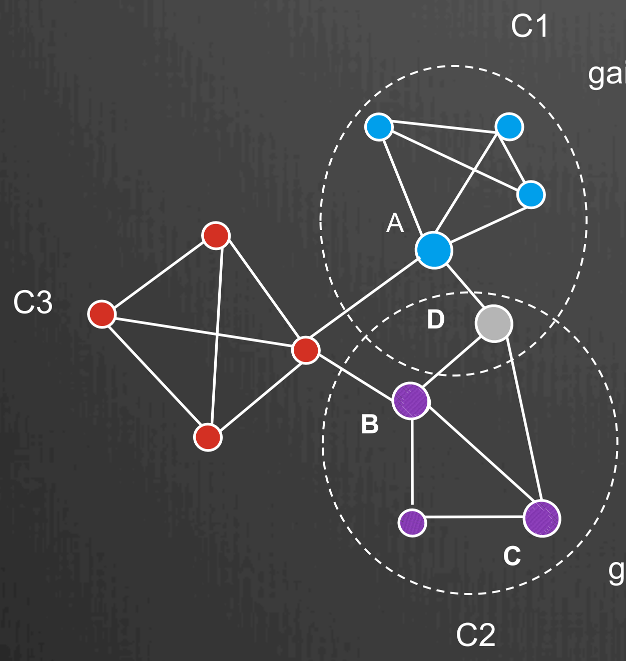




(C1 or C2) ?

```
newGraph = stockGraph.union(incGraph)
```

UMG 2 - findBestCommunity



gain1=G(node(d), community(1))

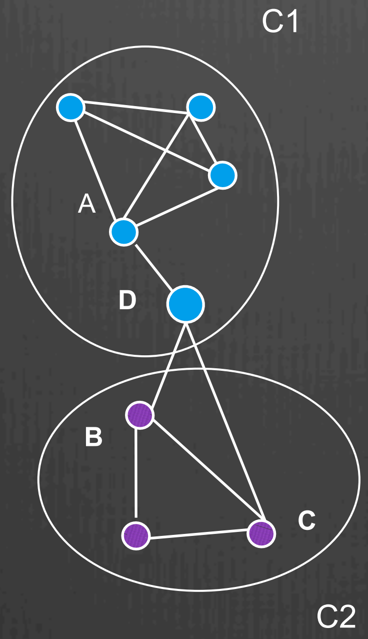
$$\Delta Q = \left[\frac{\sum in + k_{i,in}}{2m} - \left(\frac{\sum tot + k_i}{2m} \right)^2 \right] - \left[\frac{\sum in}{2m} - \left(\frac{\sum tot}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]$$

$$C_i = C \max_j G(node_i, C_j)$$

gain2=G(node(d) , community(2))

```
incVertexWithNeighbors = newGraph.mapReduceTriplets[Array[VertexData]]
(collectNeighborFunc, _ ++ _, Some((incGraph.vertices, EdgeDirection.Either)))


idCommunity = incVertexWithNeighbors.map {
  case (vid, neighbors) => (vid, findBestCommunity(neighbors))
}.cache()
```

UMG 3 - updateCommunities



$$Q = \sum_{i}^{c} Q_i = \sum_{i}^{c} \left[\frac{\sum in}{2m} - \left(\frac{\sum tot}{2m} \right)^2 \right]$$

(Q1, Q2)

```
newCommunityRdd = idCommunity.updateCommunities(commuitiyRdd)

newModularity = newCommunityRdd.map(community=>community.modularity).reduce(_+_)
```

Flow Example Code

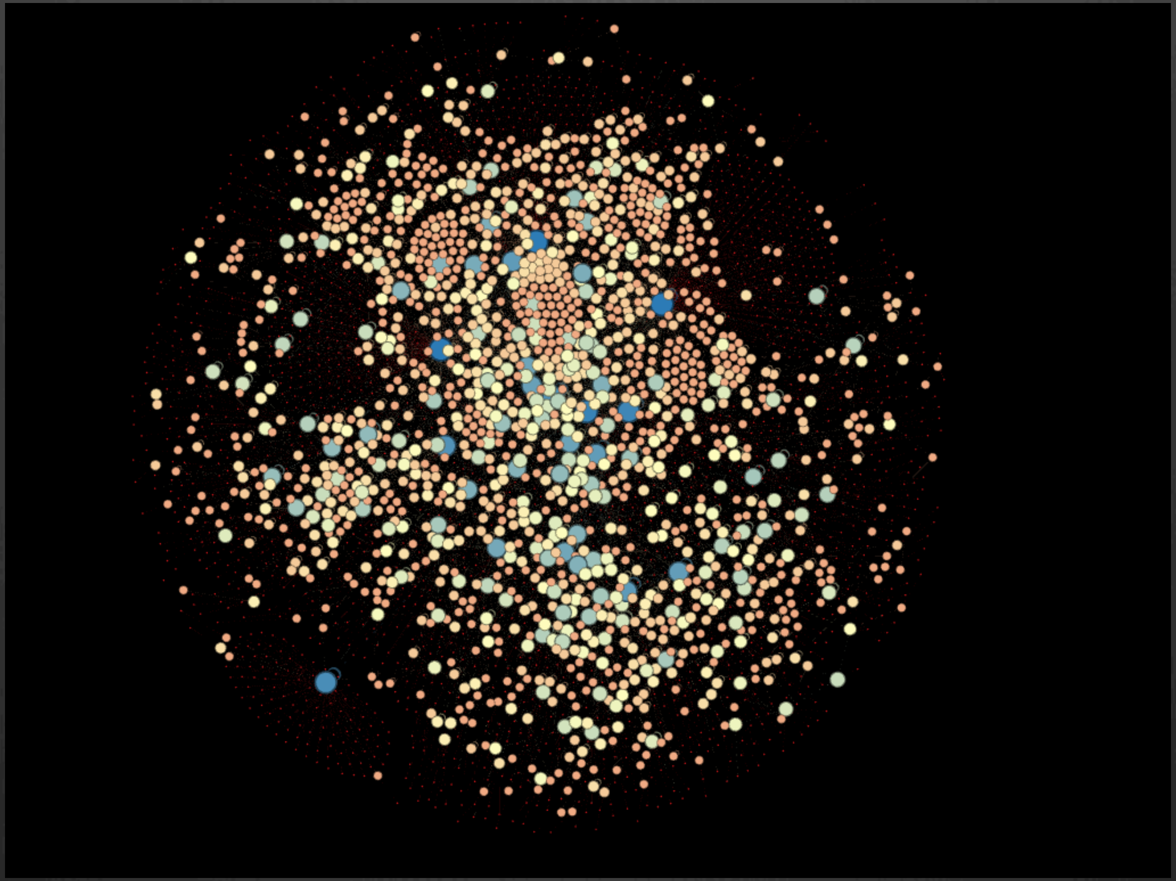
```
val conf = new SparkConf().setMaster(……).setAppName(……)
val ssc = new StreamingContext(conf, Seconds(60))


val totalGraph = initGraph(totalEdgesRdd)
Val streamingFU = new StreamingFU().setTotalGraph(totalGraph)

val onlineDataFlow = getDataFlow(ssc.sparkContext)
val edgeStreamRDD  = ssc.queueStream(onlineDataFlow, true)
```

```
edgeStreamRDD.foreachRDD {
  incEdgeRdd => {
   val incGraph  = buildIncGraph(incEdgeRdd)
   (communityInfoRDD, modularity) = streamingFU.trainOn(incGraph)
   outputToHBase(communityInfoRDD)
   outputToHBase(modularity)
   edgeRdd
  }
}
```

```
ssc.start()
ssc.awaitTermination()
```

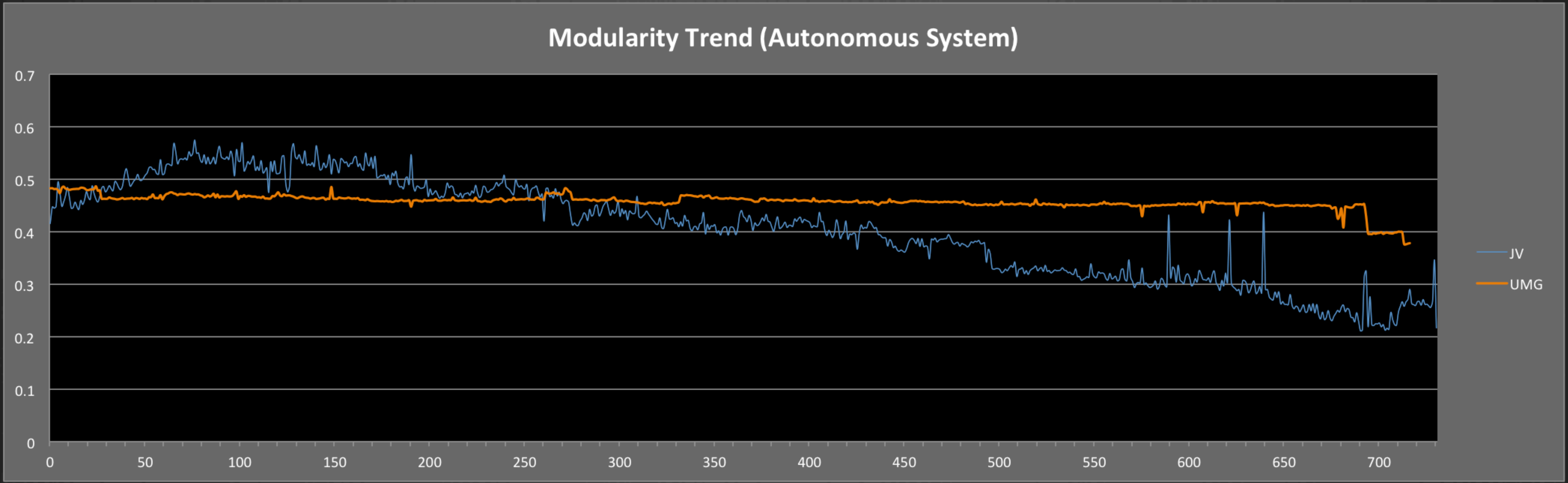
Experiment Results

Autonomous Systems Graphs

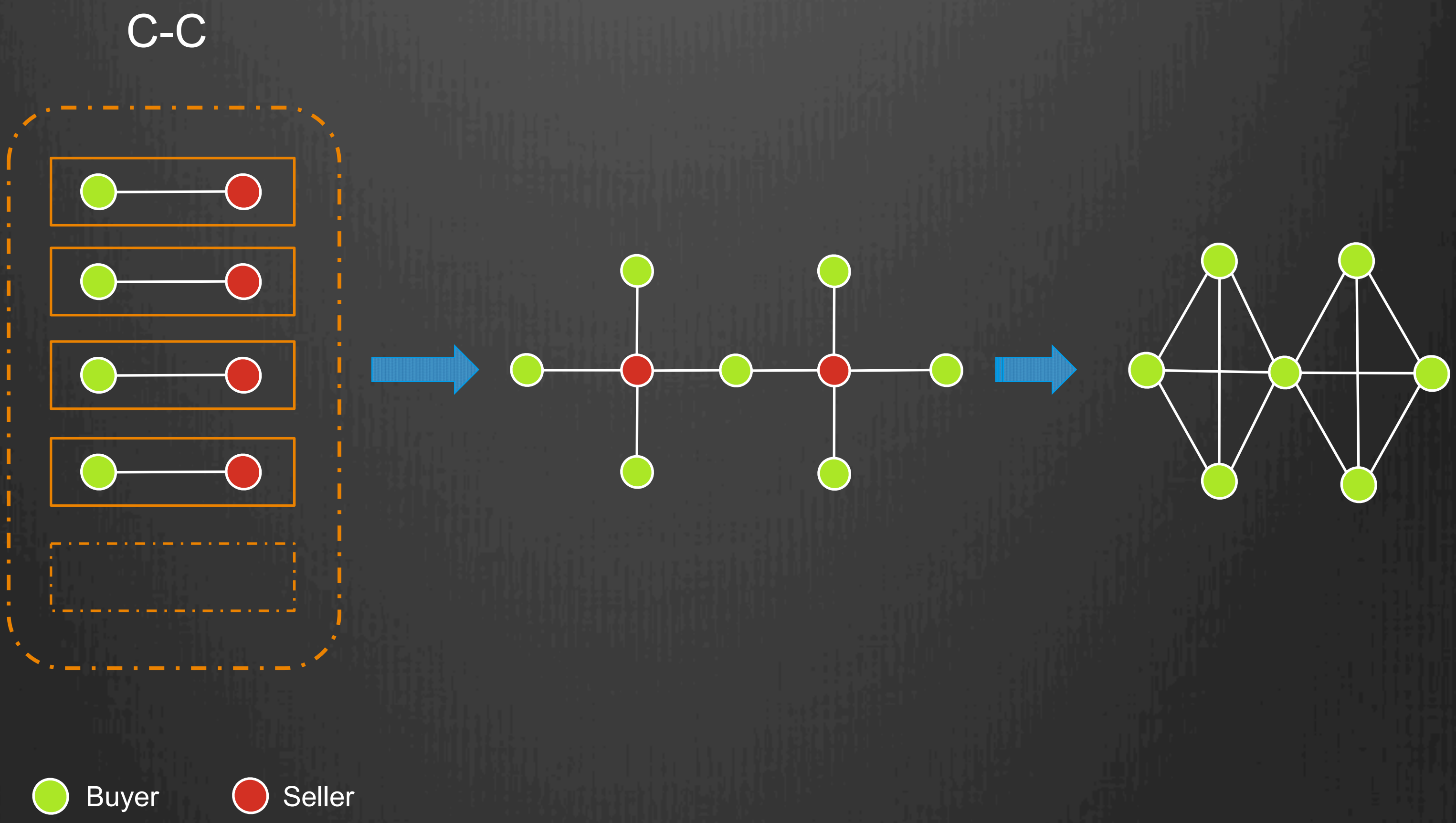


Stanford Large Network Dataset Collection(as-733)
https://snap.stanford.edu/data/

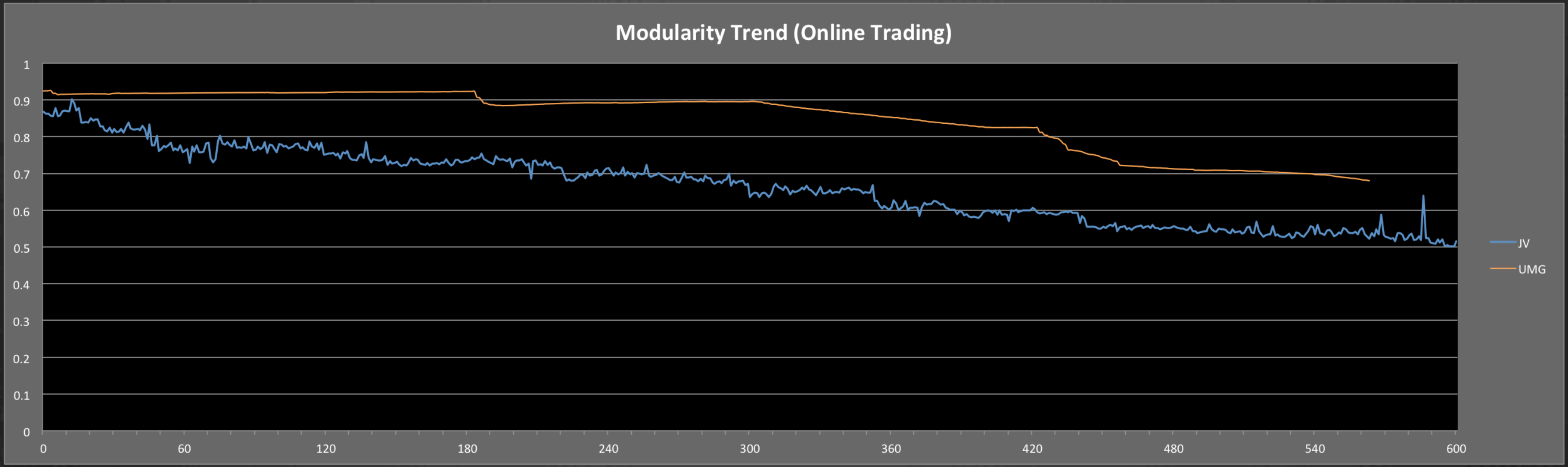
Modularity Trend – AS



Online Trading Graph



Modularity Trend – OT



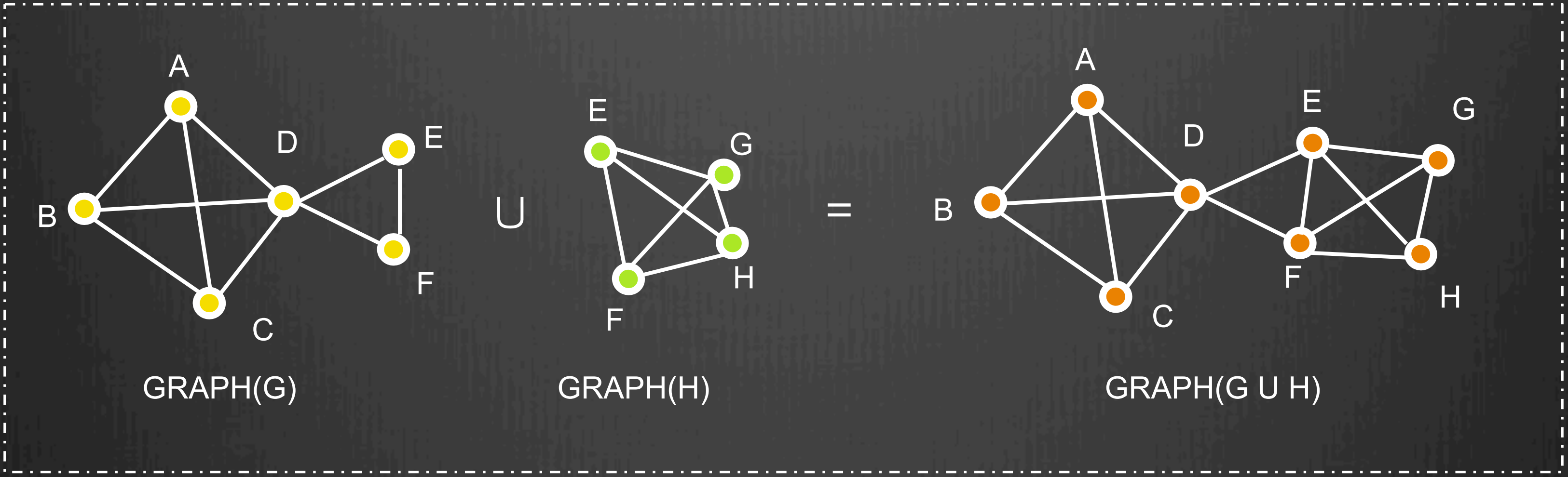
Streaming Graph → Better Result

Key Points

- Operator
  - Merge Small graph into Large graph
- Model
  - Local changes
  - Index or summary
- Algorithm
  - Delicate formula
  - Commutative law & Associative law
  - Parallelly & Incrementally

Complex GraphX Operators

Graph Union Operator



Graph Union Operator
https://issues.apache.org/jira/browse/SPARK-7894

[GraphX] Complex Operators between Graphs: Union
https://github.com/apache/spark/pull/6685

```
newGraph = stockGraph.union(incGraph)
```

Complex GraphX Operators

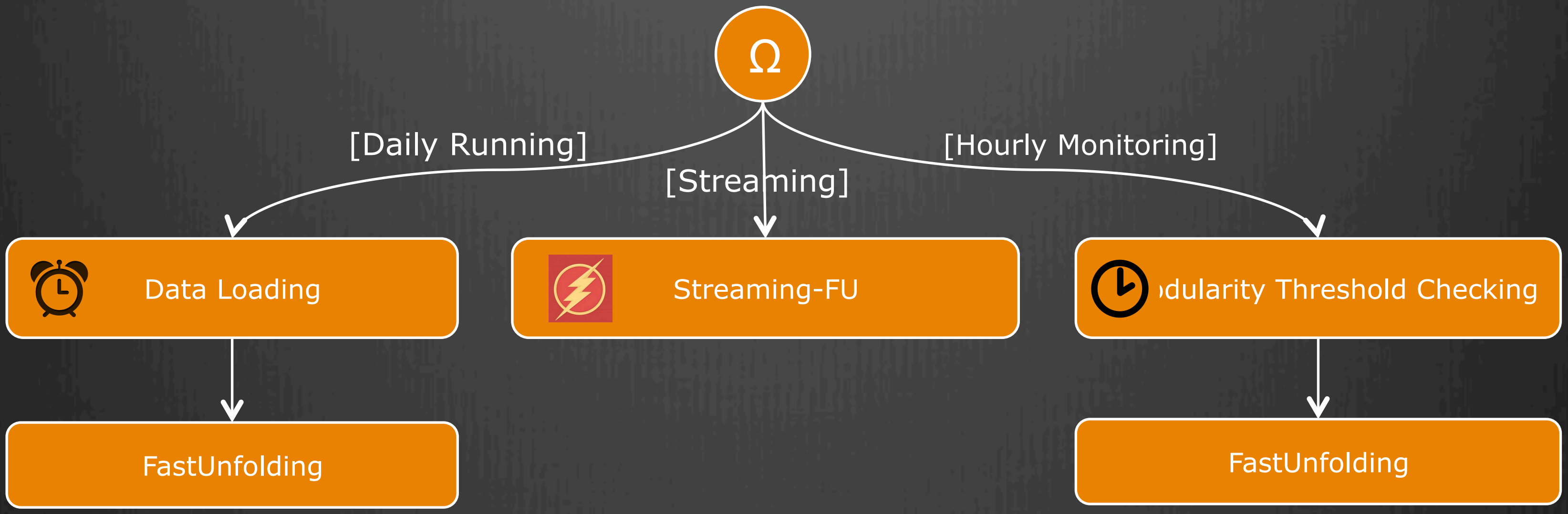
- Union of Graphs (G ∪ H)
- Intersection of Graphs (G ∩ H)
- Graph Join
- Difference of Graphs（G – H）
- Graph Complement
- Line Graph (L(G))

Issues:

Complex Operators between Graphs
https://issues.apache.org/jira/browse/SPARK-7893

Streaming Optimization

Monitoring and Correction



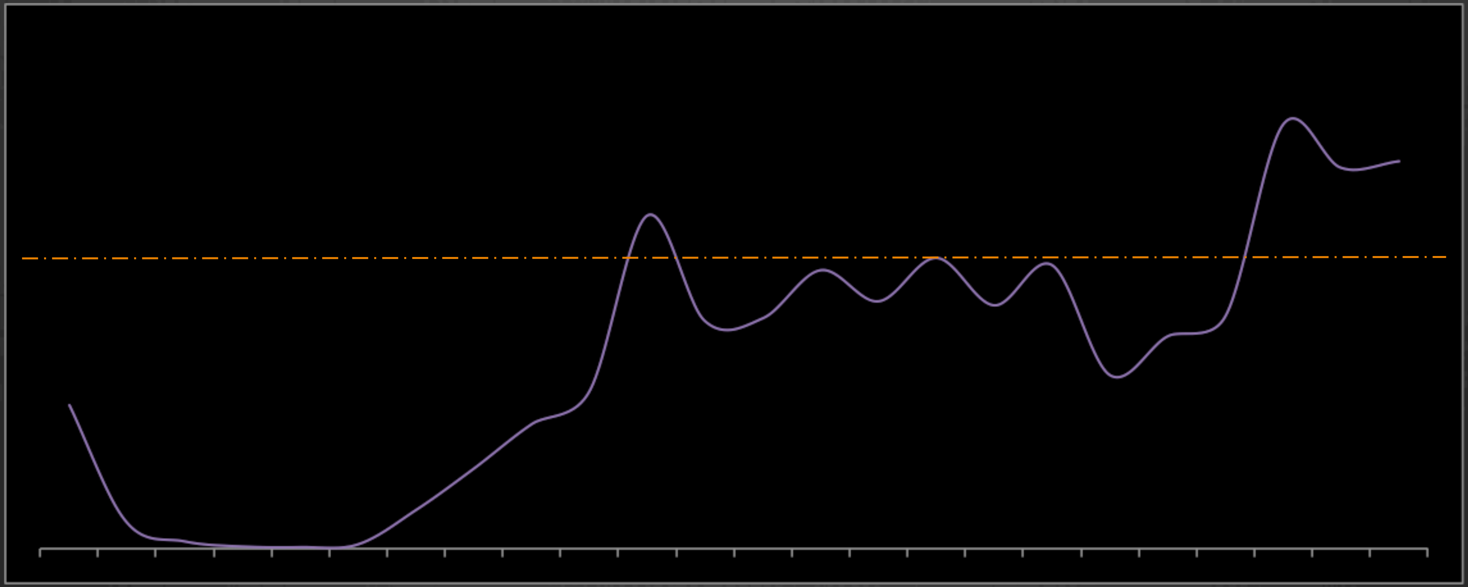
commRDDTable

communityID	**communityInfo**
community1	(in1,tot1,degree1,modularity1)
......	......

modularityTable

mTime	**mValue**
timestamp1	totalModularity1
......	......

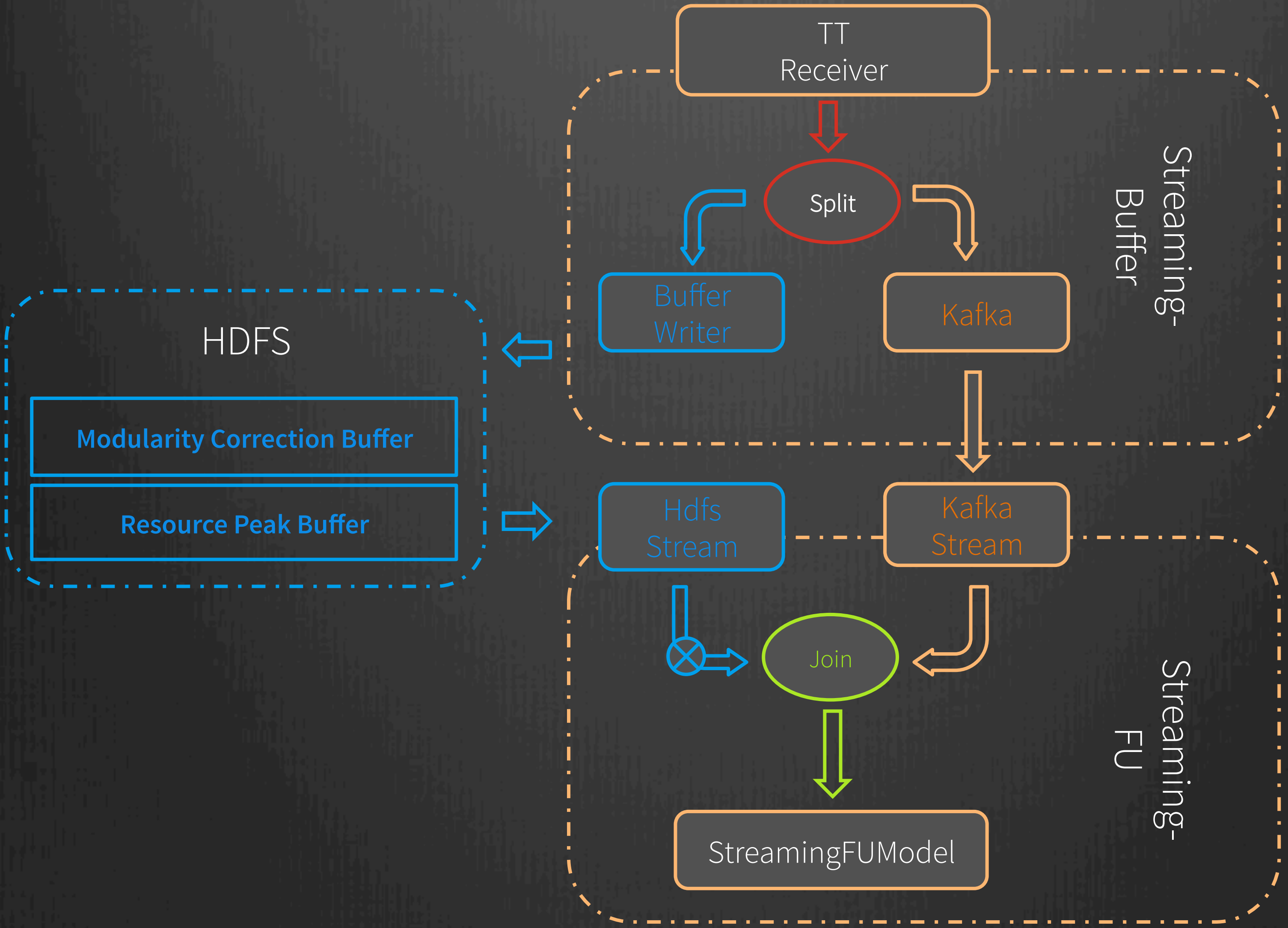
Streaming Resource Allocation



Driver-Memory：	20G
Executors：	100
Core：	2
Executor-Memory：	20G

Not Enough for Peak Period！

Streaming Buffer



Conclusion

- Streaming Graph
  - Complex Operators will help
  - Daily Rebuild & Threshold Check
  - Costs more memory and time
- Open Question

  checkpoint with Streaming or Graph?

Acknowledgements

1. Limits of community detection
   - http://www.slideshare.net/vtraag/comm-detect
2. Community Detection
   - http://www.traag.net/projects/community-detection/
3. Social Network Analysis
   - http://lorenzopaoliani.info/topics/
4. Community detection in complex networks using Extremal Optimization
   - http://arxiv.org/pdf/cond-mat/0501368.pdf

Q & A

Agenda

- Dynamic Community Detection
- Streaming Graph
- Models and Algorithms
- Complex GraphX Operators
- Streaming Optimization
- Conclusion

Static vs. Dynamic

Static Model

Dynamic Model