Bitcoin Ransomware Detection with Scalable Graph Machine Learning

Kevin Jung | Software Engineer | Stellargraph

May 2019

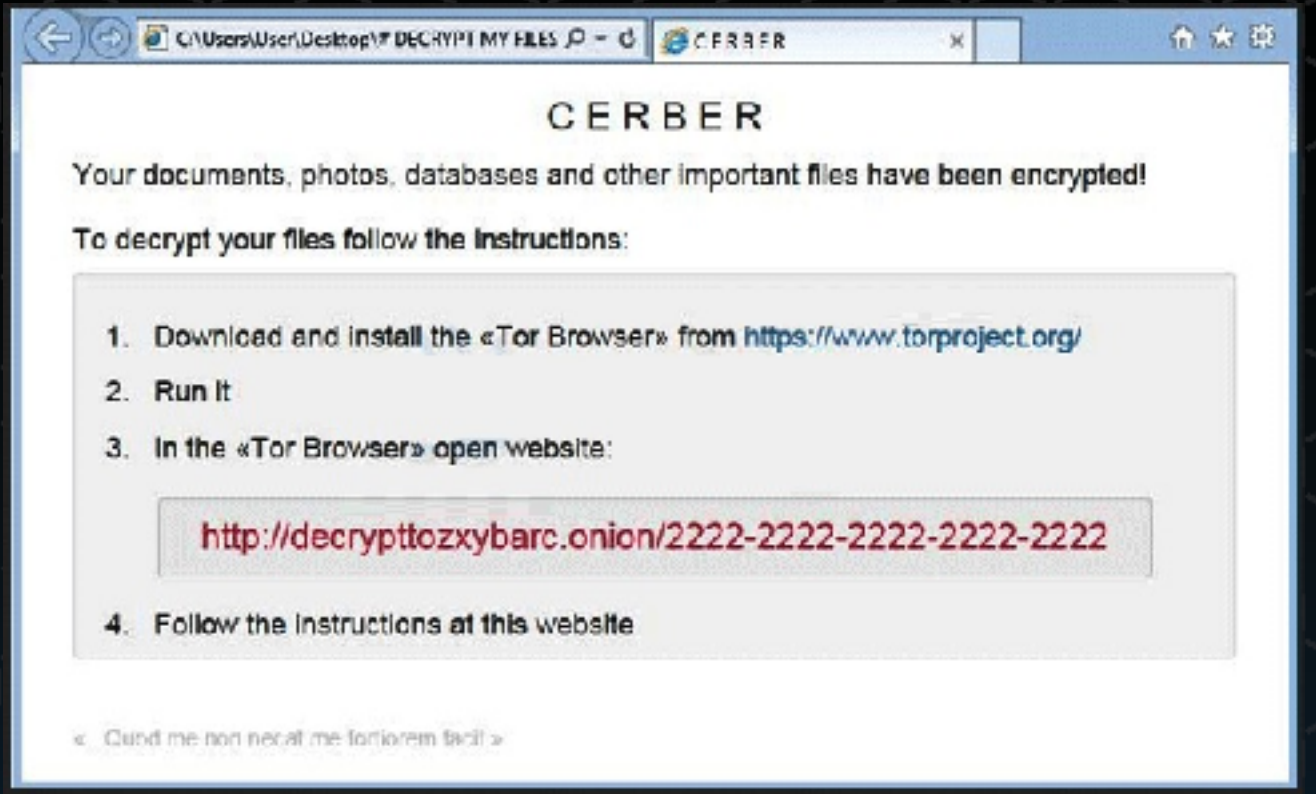
www.data61.csiro.au

Outline

- Ransomware on the Blockchain
- Graph Machine Learning
- Graph Processing in Apache Spark
- Scalability Optimisation Journey

Ransomware on the Blockchain

Ransomware

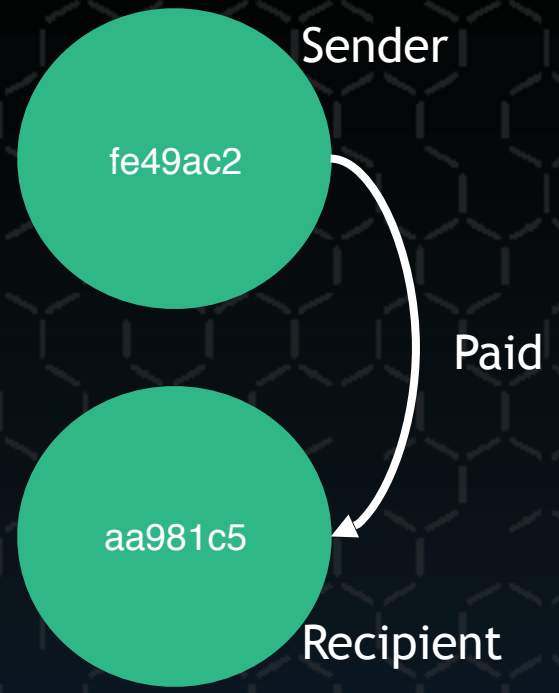

CERBER
Your documents, photos, databases and other important files have been encrypted!
To decrypt your files follow the instructions:
1. Download and install the «Tor Browser» from https://www.torproject.org/
2. Run it
3. In the «Tor Browser» open website:
http://decrypttozxybarc.onion/2222-2222-2222-2222-2222
4. Follow the instructions at this website

Detecting Ransomware Addresses

TRANSACTION			
...	IN	OUT	...
...	[fe49ac2, ...]	[aa981c5, ...]	...

Detecting Ransomware Addresses

TRANSACTION			
...	IN	OUT	...
...	[fe49ac2, ...]	[aa981c5, ...]	...



Detecting Ransomware Addresses

4 Billion Transactions
400 Million Addresses

Dataset created by Paul Rimba, Trustworthy Systems Research Group at CSIRO's Data61

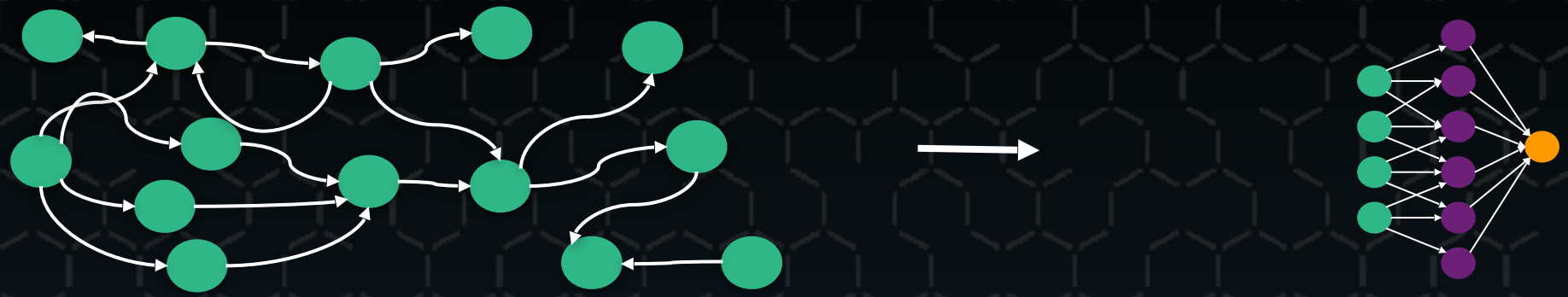
Detecting Ransomware Addresses

RANSOMWARE?

How do we detect these patterns?

An Introduction to Graph Neural Networks

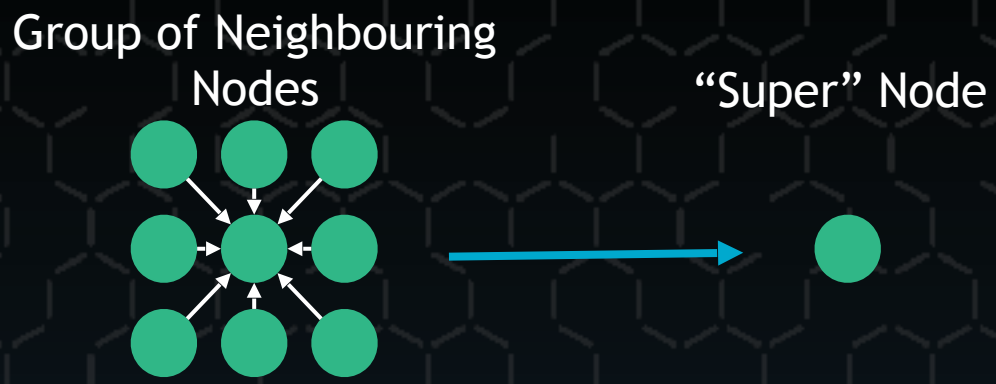
Making Use of Rich Graph Structure



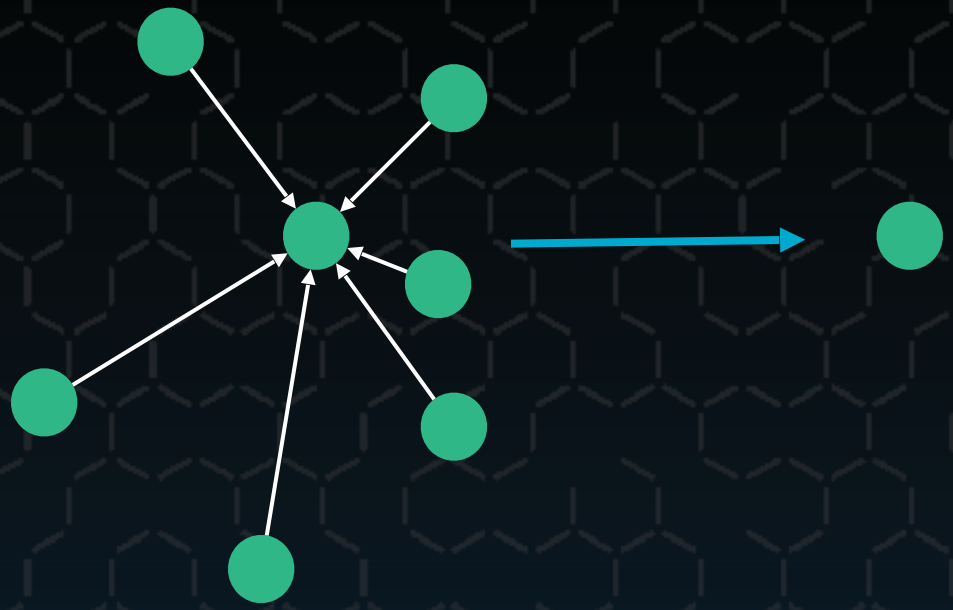
How does a neural network make use of graph structure?

Image Convolution

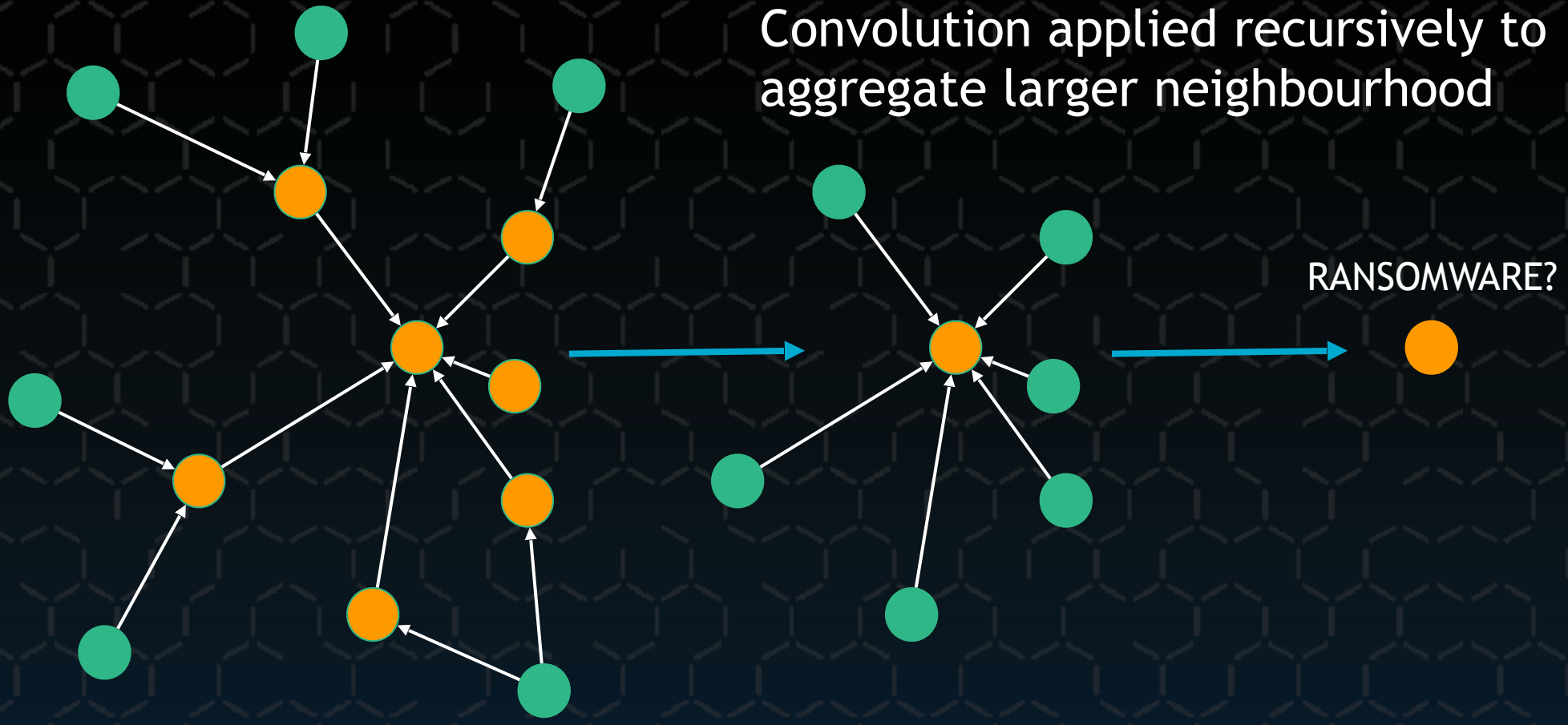
Graph Convolution



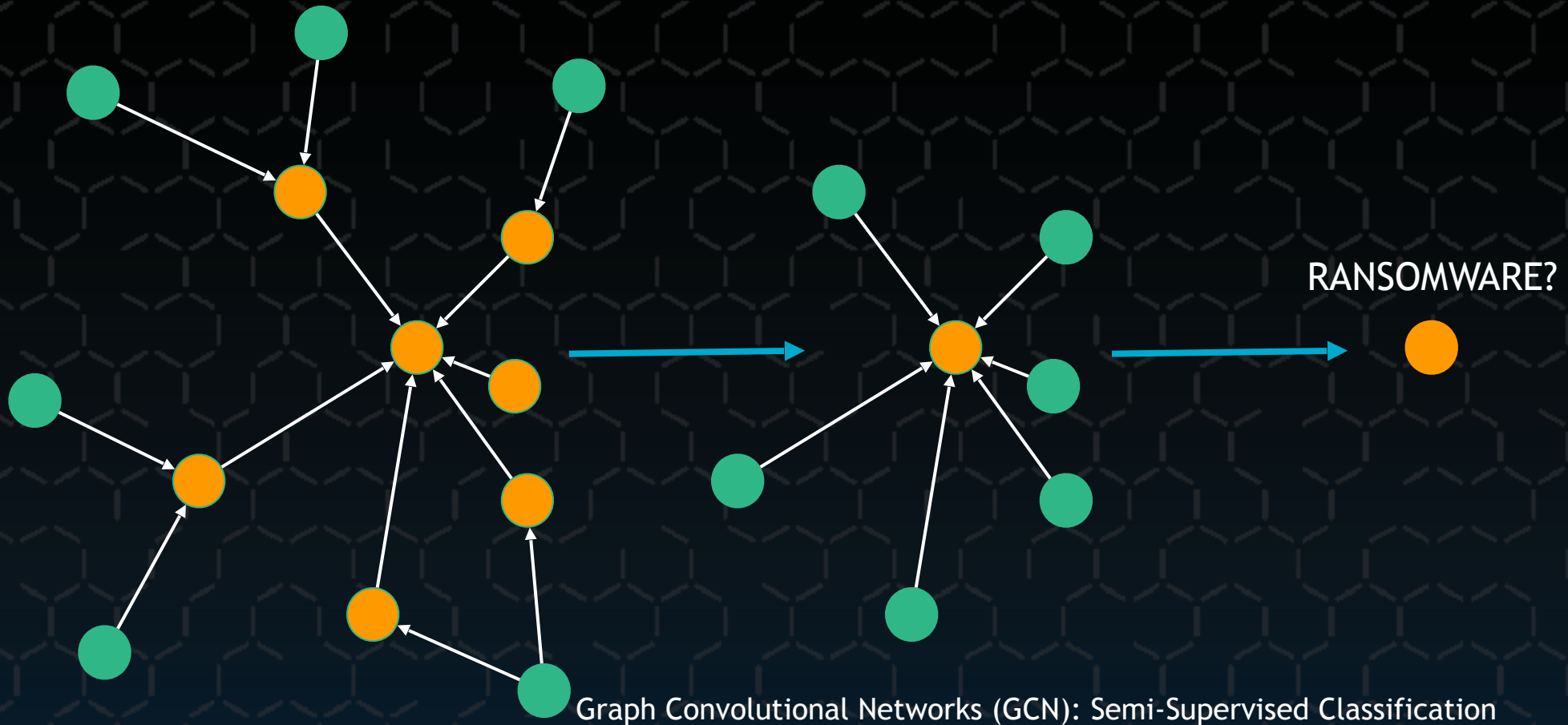
Graph Convolution



Graph Convolution

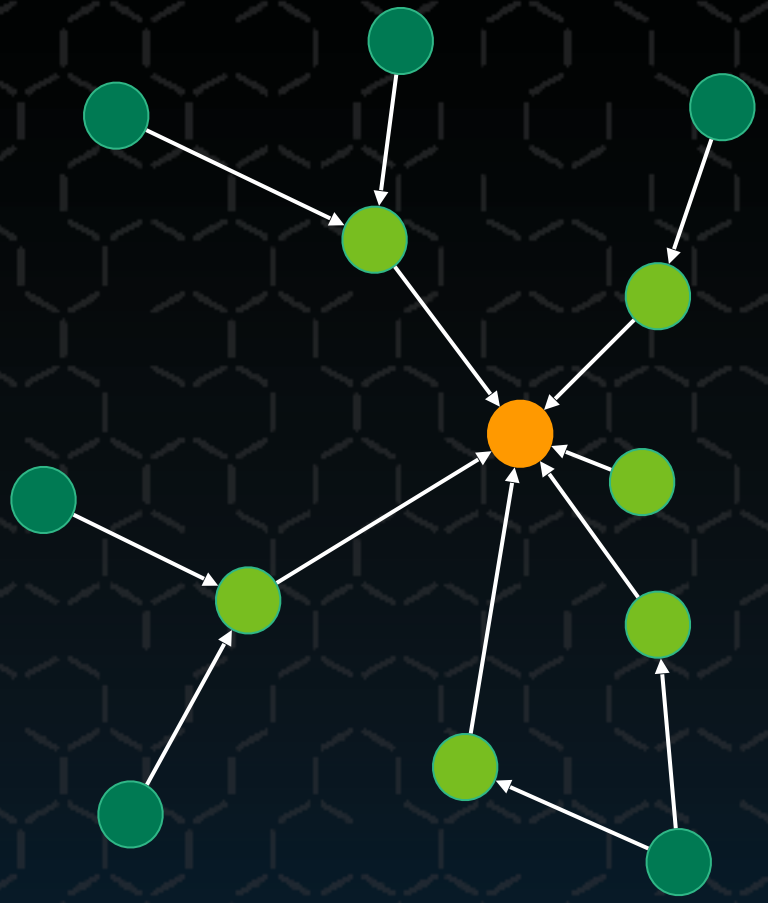


Graph Convolutional Networks (GCN)



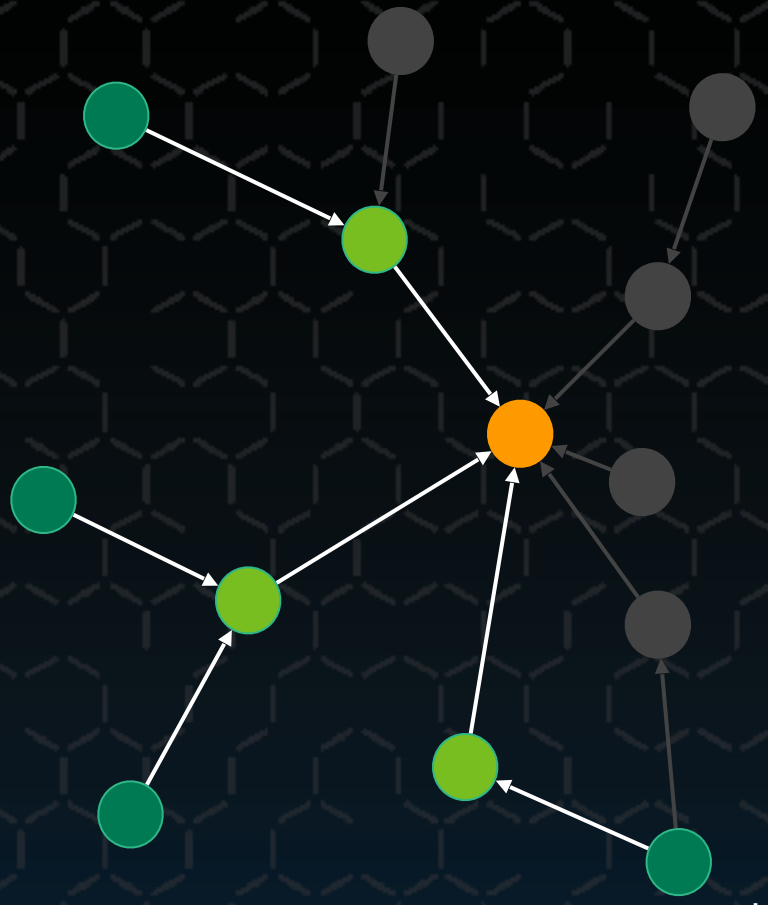
Graph Convolutional Networks (GCN): Semi-Supervised Classification with Graph Convolutional Networks. Thomas N. Kipf, Max Welling. International Conference on Learning Representations (ICLR), 2017

Graph Convolutional Networks (GCN)



- Graph structure must remain fixed
- Difficult to scale horizontally

GraphSAGE - Sample and Aggregate



- Samples a fixed-size neighbourhood for each node
- More naturally generalised to unseen data
- Easier to scale horizontally

Inductive Representation Learning on Large Graphs. W.L. Hamilton, R. Ying, and J. Leskovec arXiv:1706.02216 [cs.SI], 2017

Results

Accuracy: 0.7218

Precision: 0.8036

Recall: 0.7209

F1 Score: 0.7797

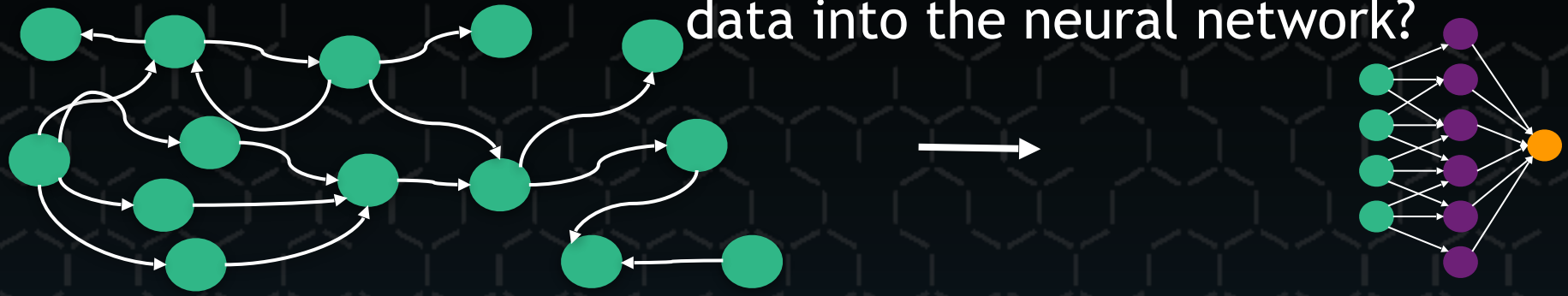
	ACTUAL RANSOMWARE	ACTUAL NON-RANSOMWARE
PREDICTED RANSOMWARE	2074	1132
PREDICTED NON-RANSOMWARE	40	967

How do we process 1B+ transactions?

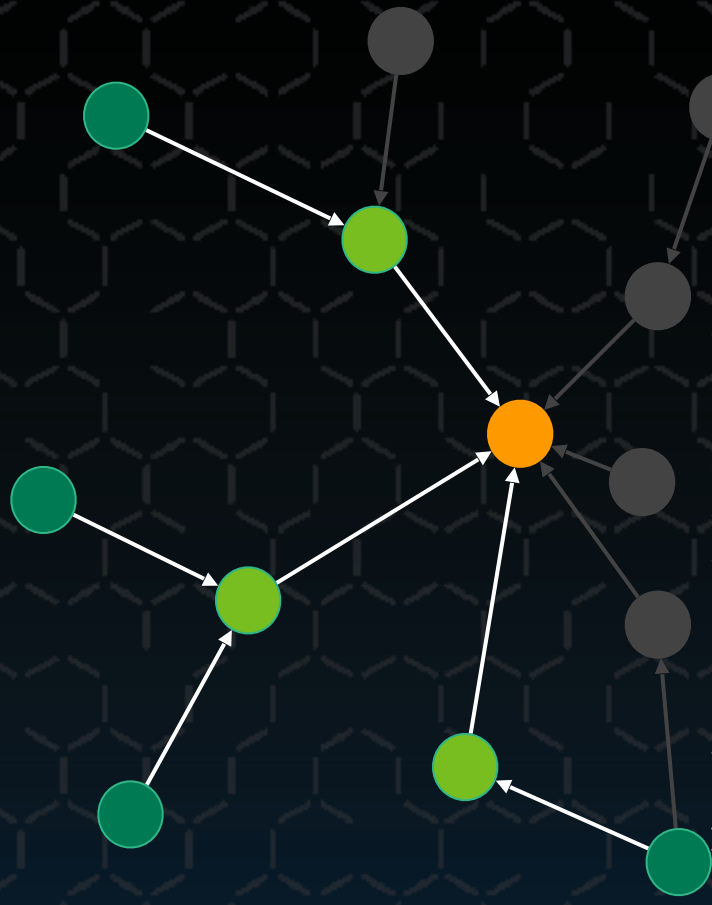
Graph Neighbourhood Aggregation in Apache Spark

Implementing the Pipeline

How do we feed our graph data into the neural network?



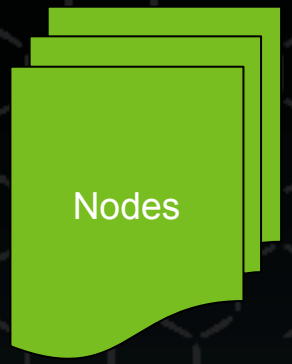
GraphSAGE in Spark



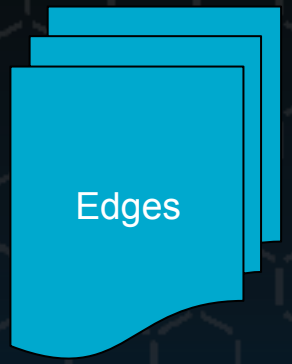
```
+------+-----------+-----------+-----------+-----
+-----------+
| node | neighbour1 | neighbour2 | neighbour3 | ... |
neighbourN |
+------+-----------+-----------+-----------+-----
+-----------+
|    a |          b |          c |          d | ... |
z |
|    b |          a |          g |          f | ... |
x |
|    c |          k |          v |          w | ... |
p |
```

Each row in this DataFrame can be used for a training iteration of GraphSAGE

Traversing the Graph in Spark

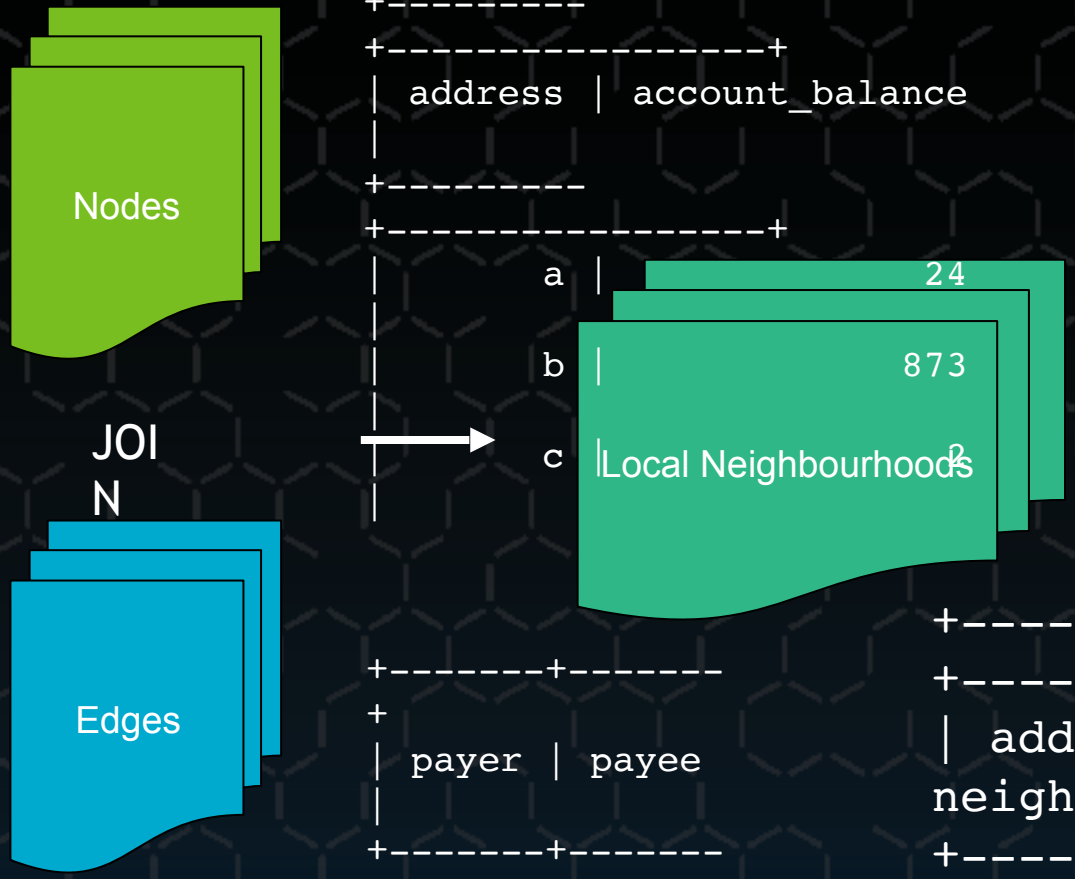



```
+---------
+------------------+
| address | account_balance
|
+---------
+------------------+
|       a |              24
|
|       b |             873
|
|       c |               2
|
```




```
+-------+-------
+
| payer | payee
|
+-------+-------
+
|     a |     b
|
|     a |     c
|
```

Traversing the Graph in Spark

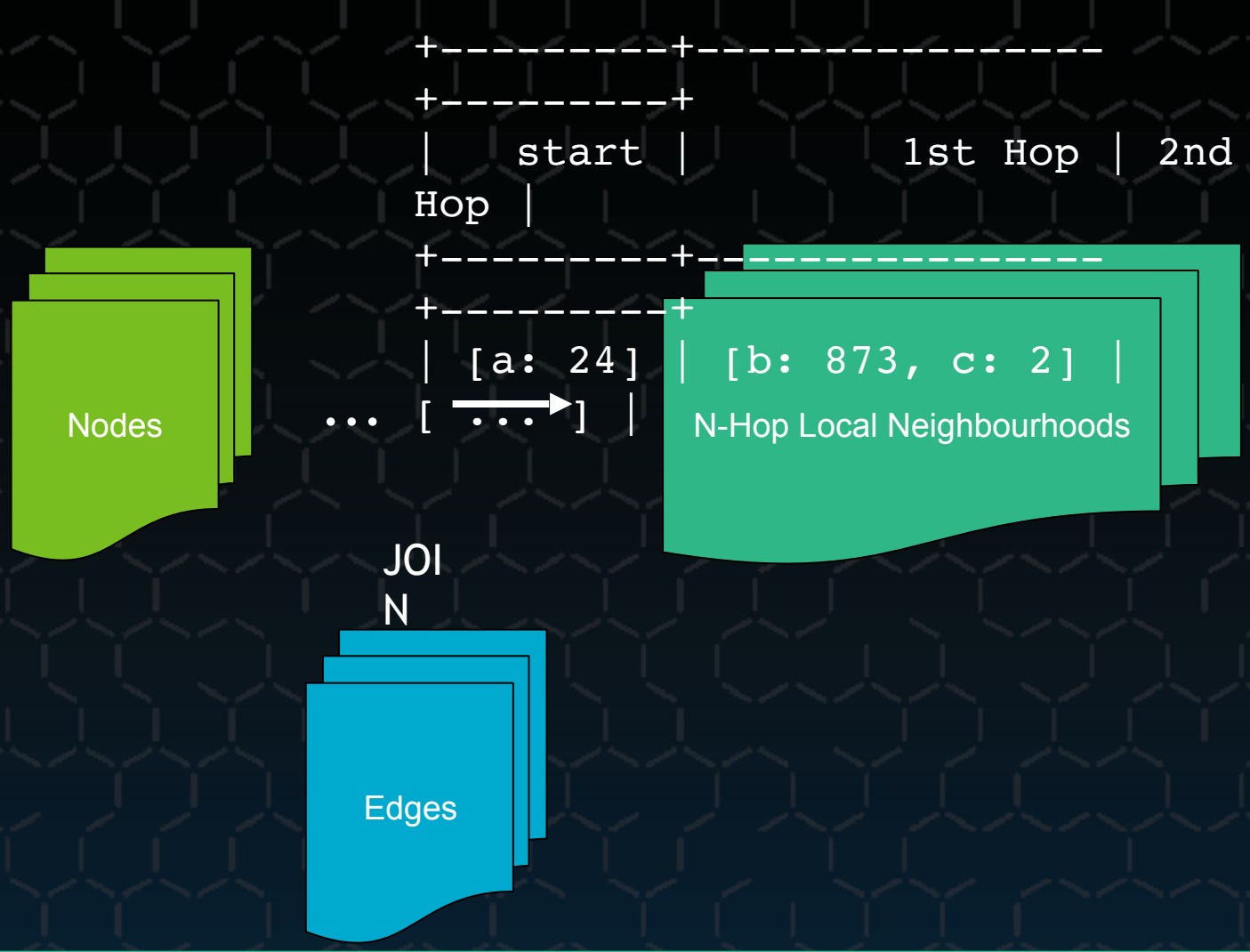


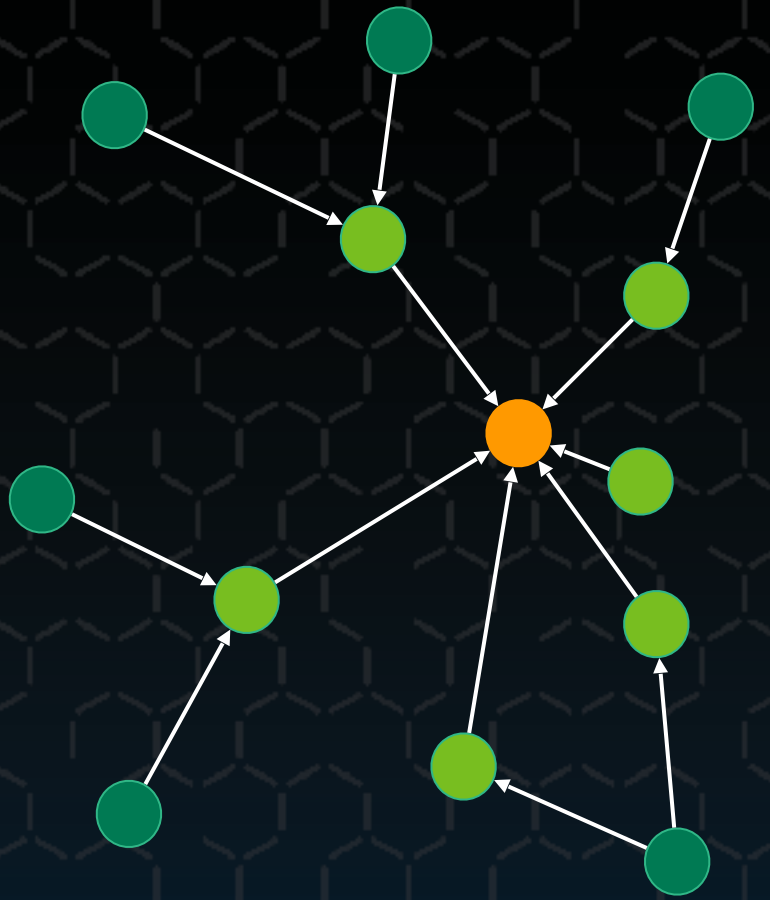

```
+---------
+-----------------+
| address | account_balance
|
+---------
+-----------------+
|       a |                 24
|
|       b |                873
|
|       c |                  2
|
```

```
+-------+-------
+
| payer | payee
|
+-------+-------
+
|     a |     b
|
|     a |     c
|
```

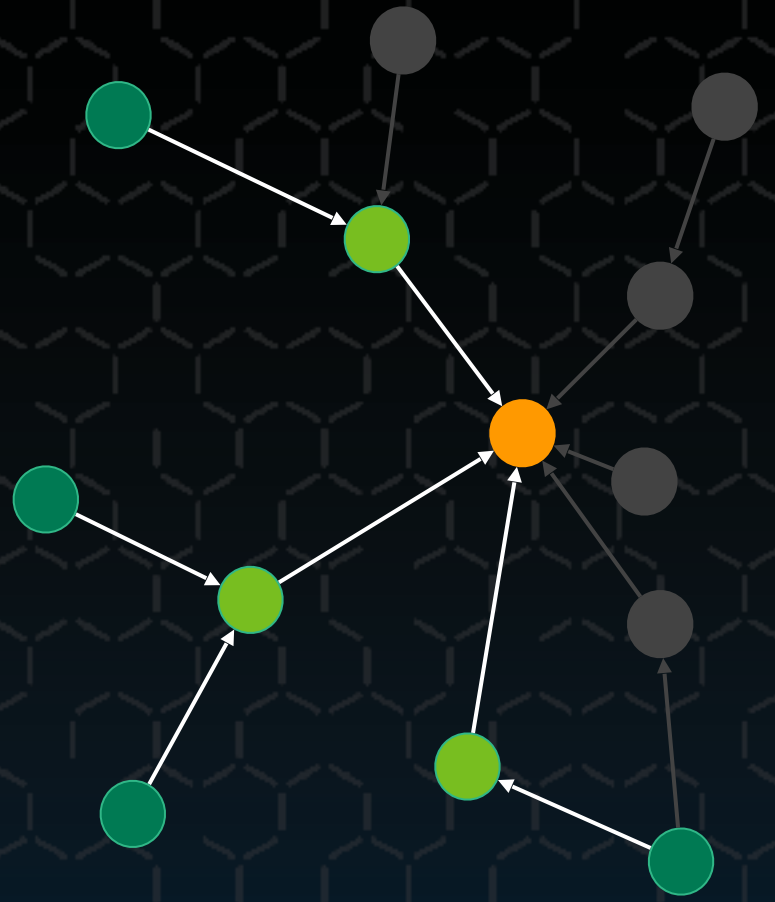
```
+---------+-----------------
+---------------+
| address | account_balance |
neighbourhood |
+---------+-----------------
+---------------+
|       a |                 24 | [b: 873, c:
2]|
```

Traversing the Graph in Spark

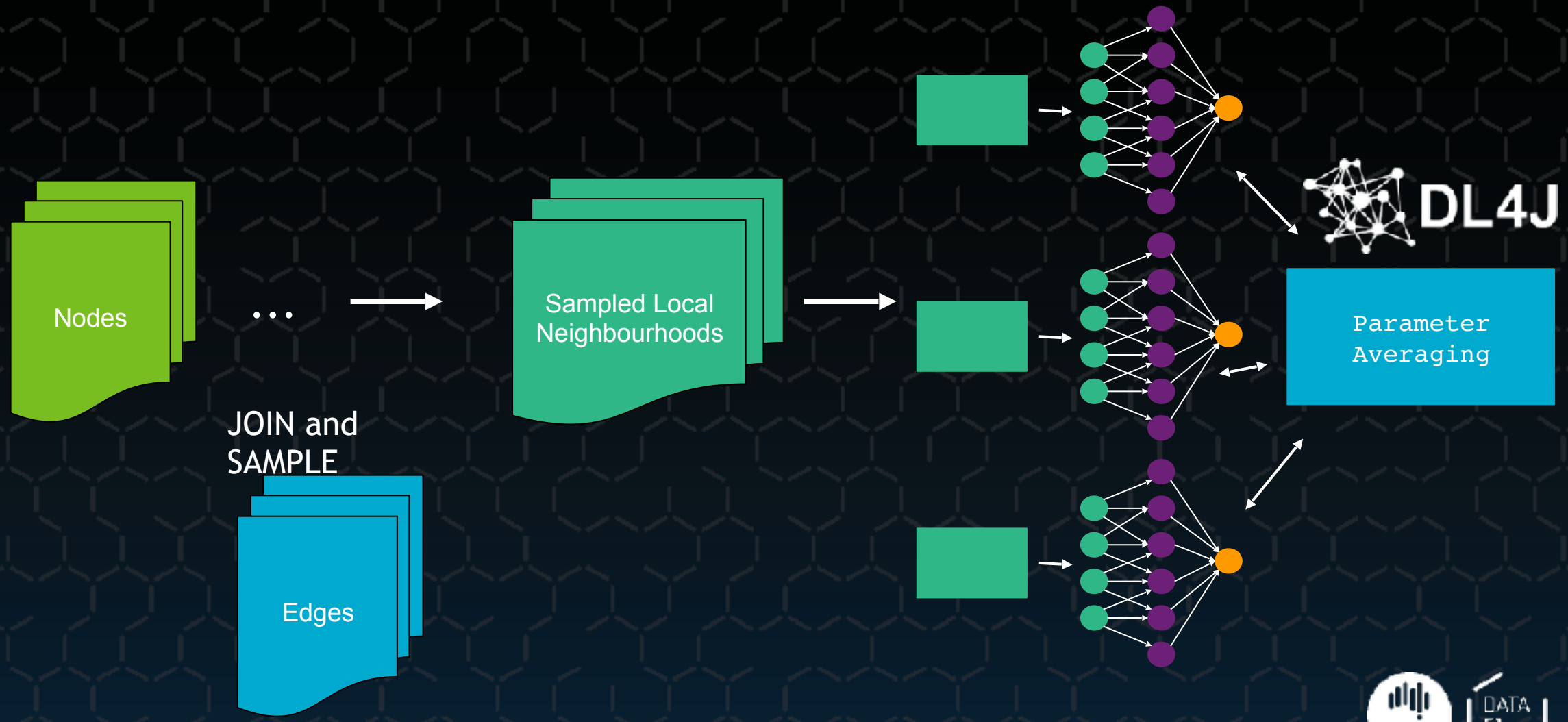




GraphSAGE in Spark - Preparing the Input

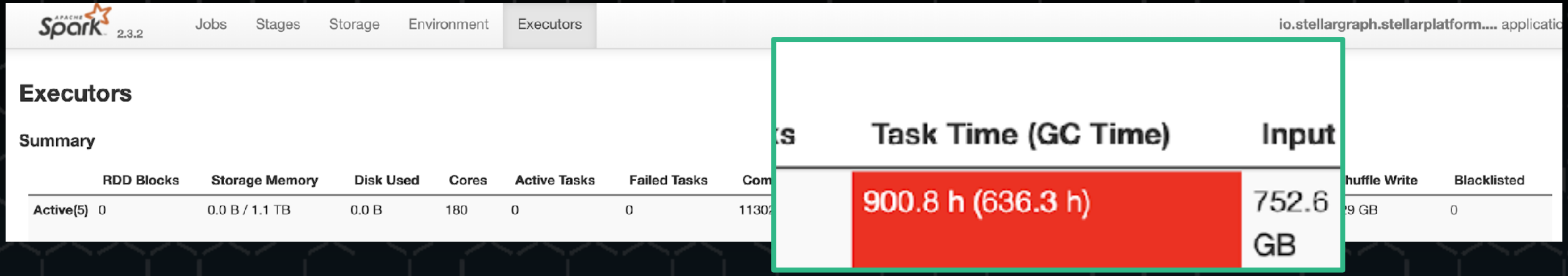


GraphSAGE in Spark - Distributed Training



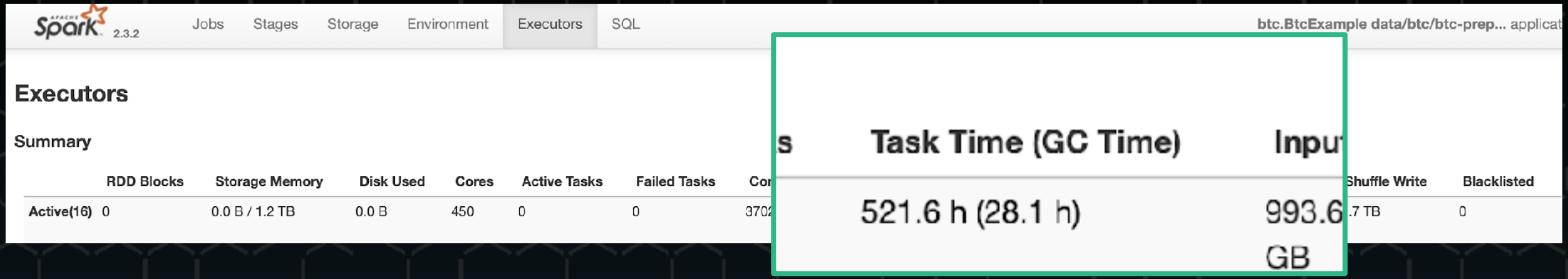
Performance Optimisation

RDD vs DataFrame



GC Time of 600 hours is taking twice as long as the remaining 300 hours of actual computation!

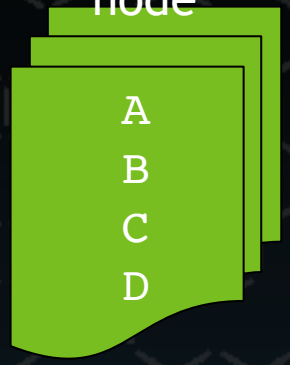
RDD vs DataFrame



DataFrames perform transformations directly on binary data stored off-heap, resulting in less GC

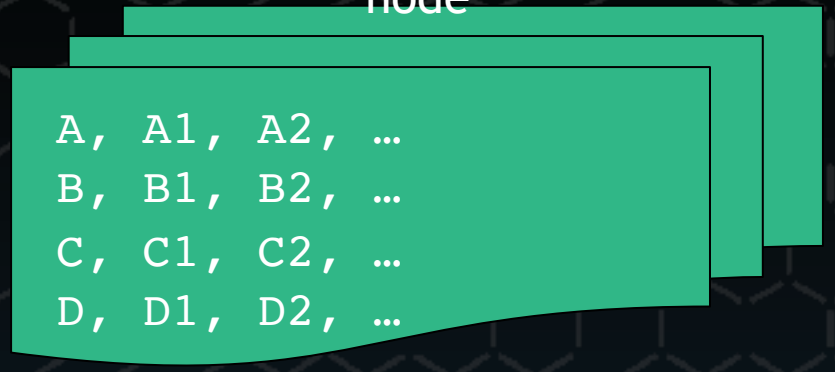
Batching DataFrames

Each row contains 1 node



...

Each row contains the neighbourhood of 1 node



May cause Out-Of-Memory Error!

Batching DataFrames

As long as we materialise the resulting splits one at a time at the end, Spark's laziness helps us avoid OOM

Hash vs Random

Hash vs Random

Sum of the parts != Total Result

Hash vs Random

Switch to deterministic randomness such as hashing unique IDs!

Scalability Results

- Cluster Specifications
  - Spark on Kubernetes
  - 8 64-core 416GB-memory machines
- Parameters
  - GraphSAGE Neighbourhood Size: 31 addresses
  - Batch Size: 100K neighbourhoods (or 3.1M addresses)
- Each batch processed (for training or prediction) in approximately 10 minutes.
- Horizontal scaling means we can expect similar processing times for larger batches by scaling out the cluster further.

Ongoing Work

- Further validation of model
- Model interpretability
  - How can we explain the prediction results by connecting them with key features of the original dataset?
  - e.g. “This node is predicted as RANSOMWARE largely because of its relationship with X and Y”
- Graph visualisation and exploration

Try it out for yourself!

github.com/stellargraph/stellargraph

We're hiring!
stellargraph.io/careers

Learn more about GraphML!
Practical Geometric Deep Learning in Python

Pantelis Elinas