GraphLab/Turi

Joshua Huseman

Background

- High performance graph-based distributed computation framework
  - Focused on large machine learning applications
- Python package
  - Written in C++ for efficiency
- Design considerations:
  - Sparse data with local dependencies
  - Iterative algorithms
  - Asynchronous execution

Timeline - confusing naming

- 2009 - Prof. Carlos Guestrin, Carnegie Mellon University, started open-source GraphLab project
- 2012 - Guestrin joins University of Washington, spins off GraphLab into its own company to support development
- 2015 - after $25 million investment, Renamed company to Dato
- 2016 - after trademark dispute with data backup company Datto, re-branded again to Turi, purchased by Apple shortly afterward for $200 million

Product line

- GraphLab Create
  - Free Academic License
  - Non-commercial use
  - Seems to have halted development after Apple acquisition
  - Install:
    - pip install --upgrade --no-cache-dir https://get.graphlab.com/GraphLab-Create/2.1/**[email]**/**[prod_key]**/GraphLab-Create-License.tar.gz
- Turi Create
  - Open-Source
  - Still in active development
  - Missing some features (i.e. Distributed Computing)
  - Install:
    - sudo apt-get install -y libblas3 liblapack3 libstdc++6 python-setuptools
    - pip install -U turicreate

Syntax

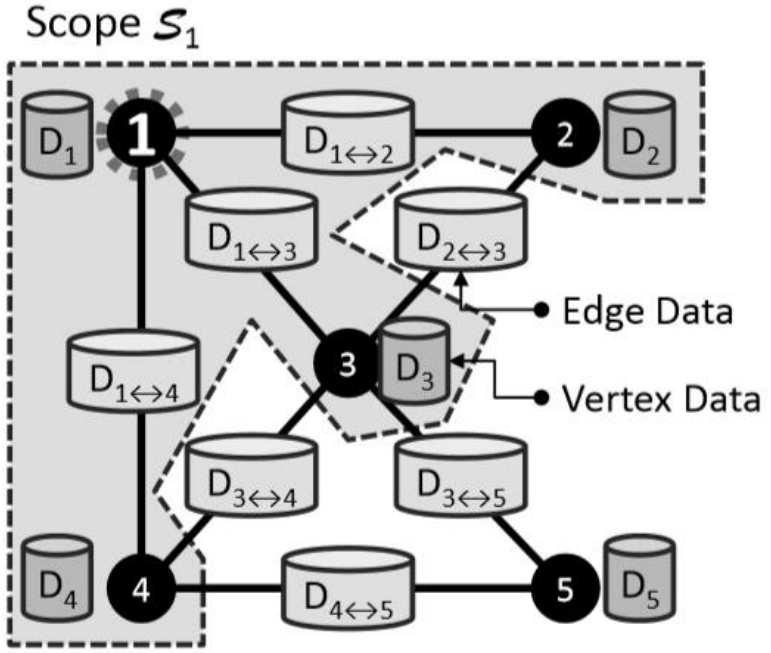
```
>>> from graphlab import SGraph, Vertex, Edge
>>> g = SGraph()
>>> verts = [Vertex(0, attr={'breed': 'Labrador'}),
>>>          Vertex(1, attr={'breed': 'Labrador'}),
>>>          Vertex(2, attr={'breed': 'vizsla'})]
>>> g = g.add_vertices(verts)
>>> g = g.add_edges(Edge(1, 2))
>>> print(g)
SGraph({'num_edges': 1, 'num_vertices': 3})
```

Data Primitives

- Persistent storage allows for data larger than system memory, in distributed storage
  - SArray
  - SFrame
  - SGraph
- Simple graph primitives:
  - Vertex
    - Vertex ID
    - Attributes – key-value pairs
  - Edge (directed)
    - Source Vertex ID
    - Destination Vertex ID
    - Attributes – key-value pairs

How are Graphs Expressed?

- Stored on distributed storage system
- “atom file” D_n stores vertices and edges of subset of graph
- “ghosts” $D_{n \leftrightarrow o}$ store edges between “atoms”
- “atom index” stores a meta-graph identifying file locations of atoms and ghosts in file system



(a) Data Graph

Distributed Execution

- Distributed jobs:
  - deploy.job.create(function)
    - Executes function asynchronously (optionally in a remote environment)
  - deploy.map_job.create(function, parameter_set)
    - Executes function asynchronously for every element in parameter_set
- Remote environment:
  - deploy.Ec2Cluster
    - Amazon EC2
  - deploy.HadoopCluster
    - Hadoop Yarn

Execution Model

Algorithm 2: GraphLab Execution Model

Input: Data Graph $G = (V, E, D)$
Input: Initial vertex set $\mathcal{T} = \{v_1, v_2, ...\}$
while $\mathcal{T}$ *is not Empty* **do**
1 $v \leftarrow$ `RemoveNext` $(\mathcal{T})$
2 $(\mathcal{T}', \mathcal{S}_v) \leftarrow f(v, \mathcal{S}_v)$
3 $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}'$
Output: Modified Data Graph $G = (V, E, D')$

- "To enable a more efficient distributed execution, we … allow the GraphLab run-time to determine the best order to execute vertices."
- Executes operations on vertices in a distributed, asynchronous queue, splitting execution across available resources

Example - remove duplicate edges

```
>>> import graphlab as gl
>>> vertices = gl.SFrame({'id':[1,2,3,4,5]})
>>> edges = gl.SFrame({'src':[1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4],
                       'dst':[2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5]})
>>> edges['edata'] = edges['src'] + edges['dst']
# Create a graph (as an example)
>>> g = gl.SGraph(vertices, edges, vid_field='id', src_field='src', dst_field='dst')
# Remove duplicates
>>> g2 = gl.SGraph(g.vertices, g.edges.groupby(['__src_id', '__dst_id'],
        {'data': gl.aggregate.SELECT_ONE('edata')}))
```

Example

```
>>> print(g.summary())
{'num_edges': 16,
'num_vertices': 5}
>>> print(g.vertices)
+------+
| __id |
+------+
|  5   |
|  2   |
|  3   |
|  1   |
|  4   |
+------+
[5 rows x 1 columns]
```

```
>>> print(g.edges)
+----------+----------+-------+
| __src_id | __dst_id | edata |
+----------+----------+-------+
|    2     |    3     |   5   |
|    2     |    3     |   5   |
|    2     |    3     |   5   |
|    2     |    3     |   5   |
|    3     |    4     |   7   |
|    3     |    4     |   7   |
+----------+----------+-------+
[16 rows x 3 columns]
Note: Only the head of the
SFrame is printed.
```

```
>>> print(g2.summary())
{'num_edges': 4,
'num_vertices': 5}
>>> print(g2.vertices)
+------+
| __id |
+------+
|  5   |
|  2   |
|  3   |
|  1   |
|  4   |
+------+
[5 rows x 1 columns]
```

```
>>> print(g2.edges)
+----------+----------+------+
| __src_id | __dst_id | data |
+----------+----------+------+
|    2     |    3     |  5   |
|    3     |    4     |  7   |
|    1     |    2     |  3   |
|    4     |    5     |  9   |
+----------+----------+------+
[4 rows x 3 columns]
```

Full API

Turi Create

Contents

- Data
  - Data Structures
    - turicreate.SArray
    - turicreate.SFrame
    - turicreate.SGraph
  - Data Types
    - turicreate.Image
    - turicreate.Vertex
    - turicreate.Edge
  - Utilities
    - turicreate.SArrayBuilder
    - turicreate.SFrameBuilder
    - turicreate.Sketch
    - turicreate.aggregate
    - turicreate.load_sframe
    - turicreate.load_sgraph
- Modelling
  - Applications
    - `activity_classifier`
    - `image_classifier`
    - `image_similarity`
    - `object_detector`
    - `recommender`
    - `text_classifier`
    - `style_transfer`
  - Essentials
    - `classifier`
    - `clustering`
    - `graph_analytics`
    - `image_analysis`
    - `nearest_neighbors`
    - `regression`
    - `text_analytics`
    - `topic_model`
  - Utilities
    - `load_model`
    - `distances`
    - `evaluation`
- Visualization
- Configuration

GraphLab Create

Contents

- Data Engineering
  - Data Structures
    - graphlab.SArray
    - graphlab.SFrame
    - graphlab.SGraph
    - graphlab.TimeSeries
  - Data Types
    - graphlab.Image
    - graphlab.Vertex
    - graphlab.Edge
  - Aggregation & Summarization
    - graphlab.Sketch
    - graphlab.aggregate
- Machine Learning
  - Machine Learning Applications
    - `anomaly_detection`
    - `churn_predictor`
    - `data_matching`
    - `deeplearning`
    - `lead_scoring`
    - `pattern_mining`
    - `recommender`
    - `sentiment_analysis`
  - Essential Machine Learning Models
    - `classifier`
    - `clustering`
    - `graph_analytics`
    - `image_analysis`
    - `nearest_neighbors`
    - `regression`
    - `text_analytics`
    - `topic_model`
  - Advanced Deep Learning with MXNet (Beta)
    - `mxnet`
    - Model Creation
  - Feature Engineering
    - `feature_engineering`
  - Model Evaluation
    - `comparison`
    - `cross_validation`
    - `evaluation`
    - `model_parameter_search`
  - Utilities and Extensions
    - `extensions`
    - `load_model`
    - `distances`
  - Distributed Model Training
    - `distributed`
- Deployment
  - Distributed/Asynchronous Execution
  - Ec2 Cluster
  - Hadoop Cluster
  - Session Management
  - Utility
- Visualization

More Resources

- https://en.wikipedia.org/wiki/GraphLab
- https://www.geekwire.com/2016/exclusive-apple-acquires-turi-major-exit-seattle-based-machine-learning-ai-startup/
- https://turi.com/index.html
- https://github.com/apple/turicreate
- https://github.com/turi-code/how-to/blob/master/remove_duplicate_edges.py
- Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin and Joseph M. Hellerstein (2012). "Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud." Proceedings of Very Large Data Bases (PVLDB).