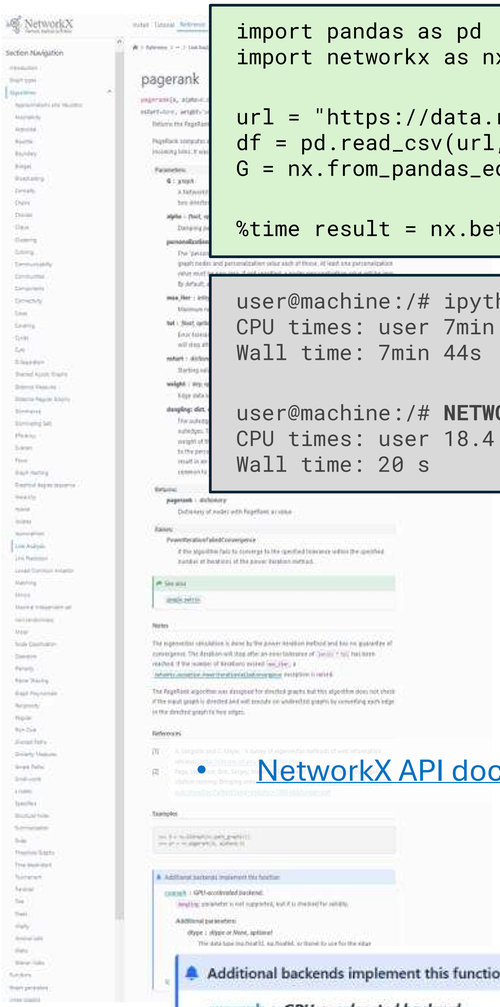
This poster

Fast and Easy Graph Analytics with the NetworkX Ecosystem of Backends

NetworkX
Network Analysis in Python

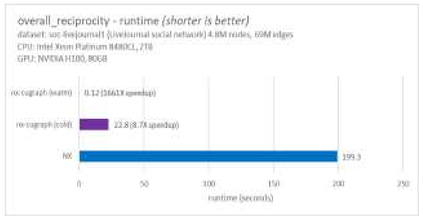
- NetworkX will automatically dispatch to **optimized implementations** provided by third-party backends, *without requiring code changes*

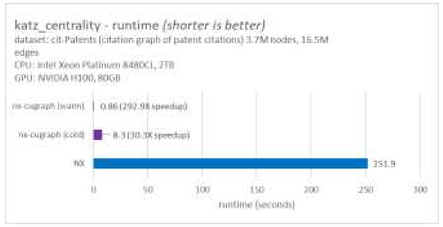


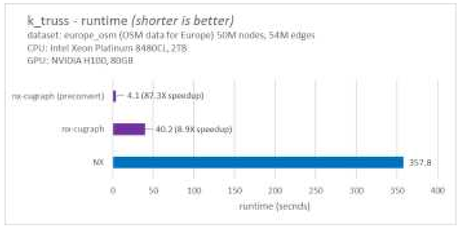
```
import pandas as pd
import networkx as nx

url = "https://data.rapids.ai/cugraph/datasets/cit-Patents.csv"
df = pd.read_csv(url, sep=" ", names=["src", "dst"], dtype="int32")
G = nx.from_pandas_edgelist(df, source="src", target="dst")

%time result = nx.betweenness_centrality(G, k=10)
```

```
user@machine:/# ipython bc_demo.ipy
CPU times: user 7min 38s, sys: 5.6 s, total: 7min 44s
Wall time: 7min 44s

user@machine:/# NETWORKX_BACKEND_PRIORITY=cugraph ipython bc_demo.ipy
CPU times: user 18.4 s, sys: 1.44 s, total: 19.9 s
Wall time: 20 s
```

- Speedup can be significant





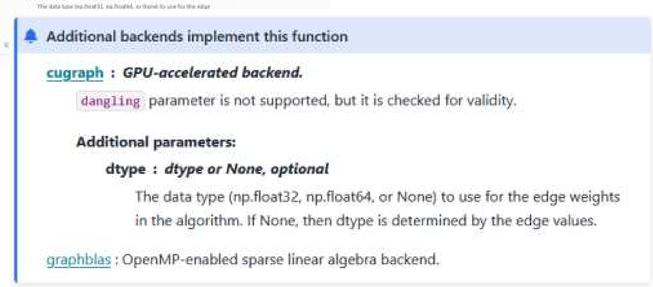


- Currently available backends allow NetworkX to:
  - run with GPU acceleration (nx-cugraph)
  - support seamless read/write access with a graph database (**nx-arangodb**)
  - parallelize algorithms on multiple CPU cores using joblib (nx-parallel)
  - use GraphBLAS implementations (graphblas-algorithms)
  - use a Pandas DataFrame as a Graph (nx-pandas)

NetworkX Network Analysis in Python + NVIDIA ArangoDB Joblib pandas

- NetworkX API docs include details on available backends

For more information, visit:
https://networkx.org/documentation/latest/reference/backends.html



- Backends can also add support for new NetworkX input types

```
import pandas as pd
import networkx as nx

df = pd.read_csv("data.csv")
nx.pagerank(df)  # columns can be specified, uses "source" and "target" by default
```

- User-configured backend priority allows for automatic failover if an algorithm is not supported
  - *Ex. cugraph → graphblas → networkx*

```
user@machine:/# NETWORKX_BACKEND_PRIORITY="cugraph,graphblas" python script.py
```

- Users have a variety of techniques to dispatch calls to backends

```
import networkx as nx

G = nx.karate_club_graph()
# forces dispatch to cugraph, error if not supported
pr_vals = nx.pagerank(G, backend="cugraph")

# backend types always dispatch to their backend, error if not supported
import nx_cugraph as nxcg
Gcg = nxcg.from_networkx(G)
pr_vals = nx.pagerank(Gcg)

# configure to dispatch to cugraph, fallback to networkx if not supported
nx.config.backend_priority = ["cugraph"]
pr_vals = nx.pagerank(G)
```

Rick Ratzel (NVIDIA), Dan Schult (Colgate University)