Title

Erik’s introduction

aditi's intro

<Scipy Poster>

- NetworkX dispatching
    - What it is? - <diagram>
    - What is this @_dispatchable decorator in the above diagram that does all of this?
        - <diagram to explain _dispatchable's workings>

nx.betweenness_centrality(G, backend="parallel")

nx.betweenness_centrality(cuG)

NetworkX

```
@_dispatchable
def betweenness_centrality(G, ...):
    ....
```

nx-parallel

```
def betweenness_centrality(G, ...):
    ....
```

cuGraphs

```
def betweenness_centrality(G, ...):
    ....
```

Graphblas

```
def betweenness_centrality(G, ...):
    ....
```

NETWORKX_AUTOMATIC_BACKEND=graphblas

nx.betweenness_centrality(G)

<@_dispatchable decorator diagram>

- What makes up a networkx backend?:
    - `entry_point`s in metadata
    - BackendInterface object -
        - convert_from_nx
        - convert_to_nx, and
        - the functions
- Backend graph_object with __networkx_backend__ attribute

<switch to tutorial(kind of)>

- create nx backend(dummy)
- ways of using of this as a user
    - backend kwarg
    - Backend graph
    - Environment variable
    - Using multiple backends together(backend_priority)
-test dummy backend

- on_start_run, can_run, should_run, get_info
- logging, configs(go on to nx-parallel)

- Nx-parallel configs system
  - About nx-parallel (chunking-if time permits)
  - Config : expectation vs reality

- Give away to Erik
- Multiverse of networkx backends

End slide