Enabling Fastai Multi-GPU/DDP Training in Jupyter

- `fastai`: to make building high-performing AI application easy
  - Open source, free lessons videos + Jupyter notebooks
- **Distributed Data Parallel (DDP)**:
  - *a multiprocess-based parallelism to speed up training with multiple GPUs/nodes.*

"`fastai` + DDP" cmdline app is trivial [1], but ...

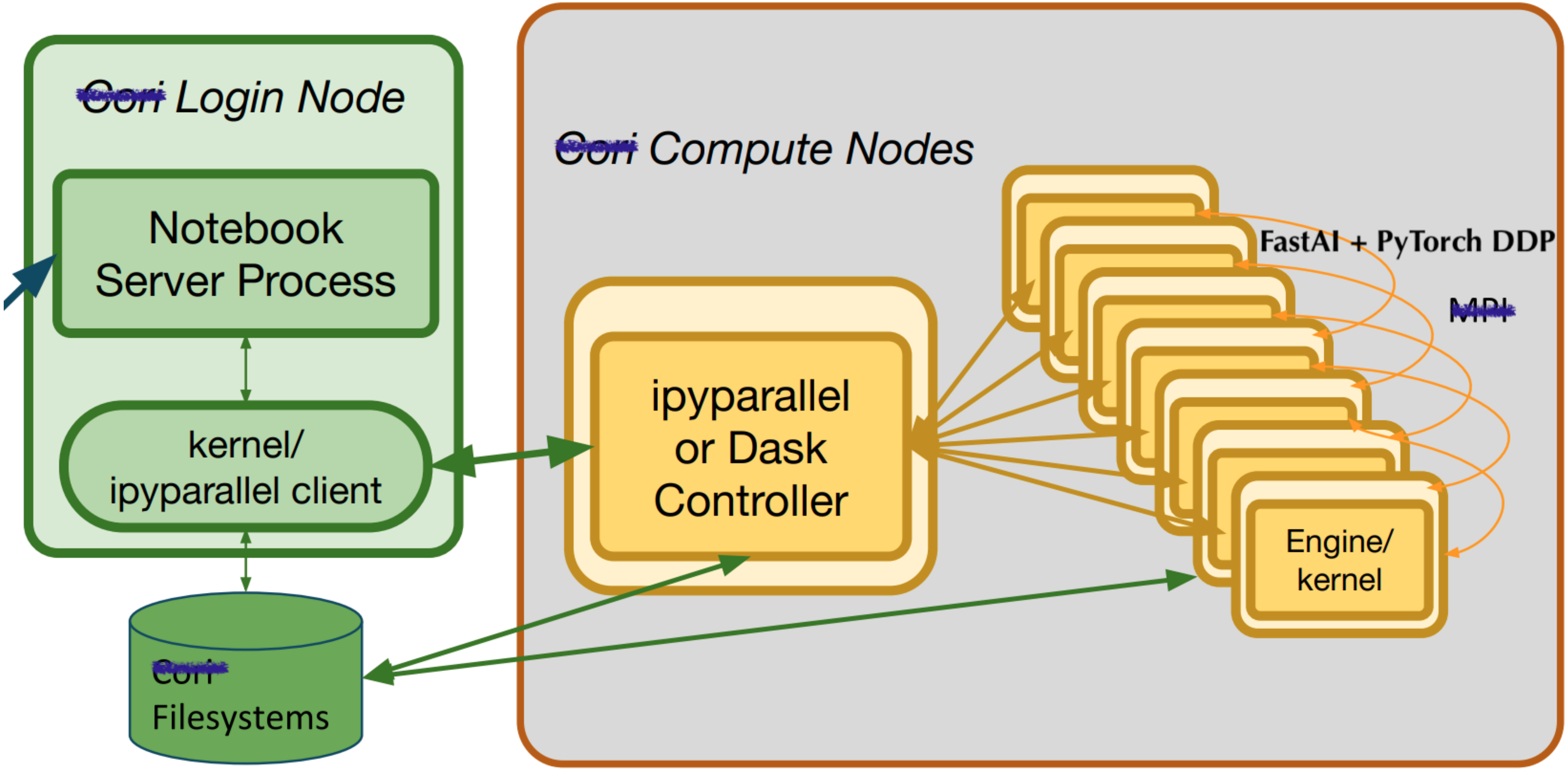
> > ***"Distributed training* doesn't *work in a notebook..."*[2]**

Ddip tries to bridge this gap.

[1] See Reproducing DAWNBench winning results in a few lines of code
[2] *FastAI's tutorial on How to launch a distributed training*
[3] A neat diagram of the DDP architecture from this blog.

`Ddip` - Distributed data "interactive" parallel

An iPython extension to control PyTorch DDP from within Jupyter, uses `ipyparallel` underneath.



ce of
nce

Diagram borrowed from a presentation by NERSC/LBNL at https://bluewaters.ncsa.illinois.edu/liferay-content/document-library/content/Blue%20Waters%20Webinar.%20Jupyter%20as%20the%20Interface%20to%20High-Performance%20Computing.%20Thomas%20&%20Cholia.pdf

`Ddip` - Distributed data "interactive" parallel

Usage:

Control DDP and cell execution destination using `%` and `%%` magics

- `%load_ext Ddip`, to load the extension.
- `%makedip ...`, to start/stop/restart a DDP group, and an app, e.g. `fastai_v1`.
- `%%dip {remote, local, everywhere} ...`, where to execute the cell.
- `%autodip {on,off}`, to automatically prepend `%%dip` to subsequent cells.
- `%dipush`, and `%dipull`, to pass objects between the notebook and the DDP namespaces.

Speedup in Training

Notebook	[3-GPUs timing]	[Single-GPU timing]
lesson3-CamVid:	[3:30,4:24,12:00,12:52]	[7:33,9:12,31:50,33:40]
lesson3-planet:	[3:20,3:45,6:15,7:30]	[4:20,5:35,14:35,18:30]
lesson7-superres-imagenet:	[4:17]	[10:50]
lesson7-wgan:	[13:30/epoch] ***Ouch***!	[4:41/epoch]

Limitations

- Works on a single node with multiple GPUs only. Luckily `ipyparallel` does support multiple nodes.
- Not all models gain speedup in training: one model is flat, one has accuracy problem, ***a wgan model manages to achieve linear slow down!!*** 😭

`Ddip` - Distributed data "interactive" parallel

Fun Lessons learned:

1. Python's dynamic nature empowers: patch classes dynamically, toss objects/functions across multiple processes.
2. Hooks/callbacks architecture are truly flexible: Jupyter, `fastai`.
3. Multiprocess + multi-GPUs + Jupyter offer interesting complexity and opportunities
   - data movement, race conditions, proc & mem mgmt.
4. Design choices:
   - *user semantics -* `%`, `%%` *vs library calls*
   - *deadlock solutions - single proc or careful synchronization*
   - *implicit vs explicit: what to automate, what to display*

Looking Forward

- `fastai v2` is out already.
- `nbdev` as productivity boost.
- Support cluster of nodes.
- Feedbacks and contribution via github are welcome! (*philtrade@winphil.net*)

Github Repo: https://github.com/philtrade/Ddip/