

# 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 multiprocessing-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]

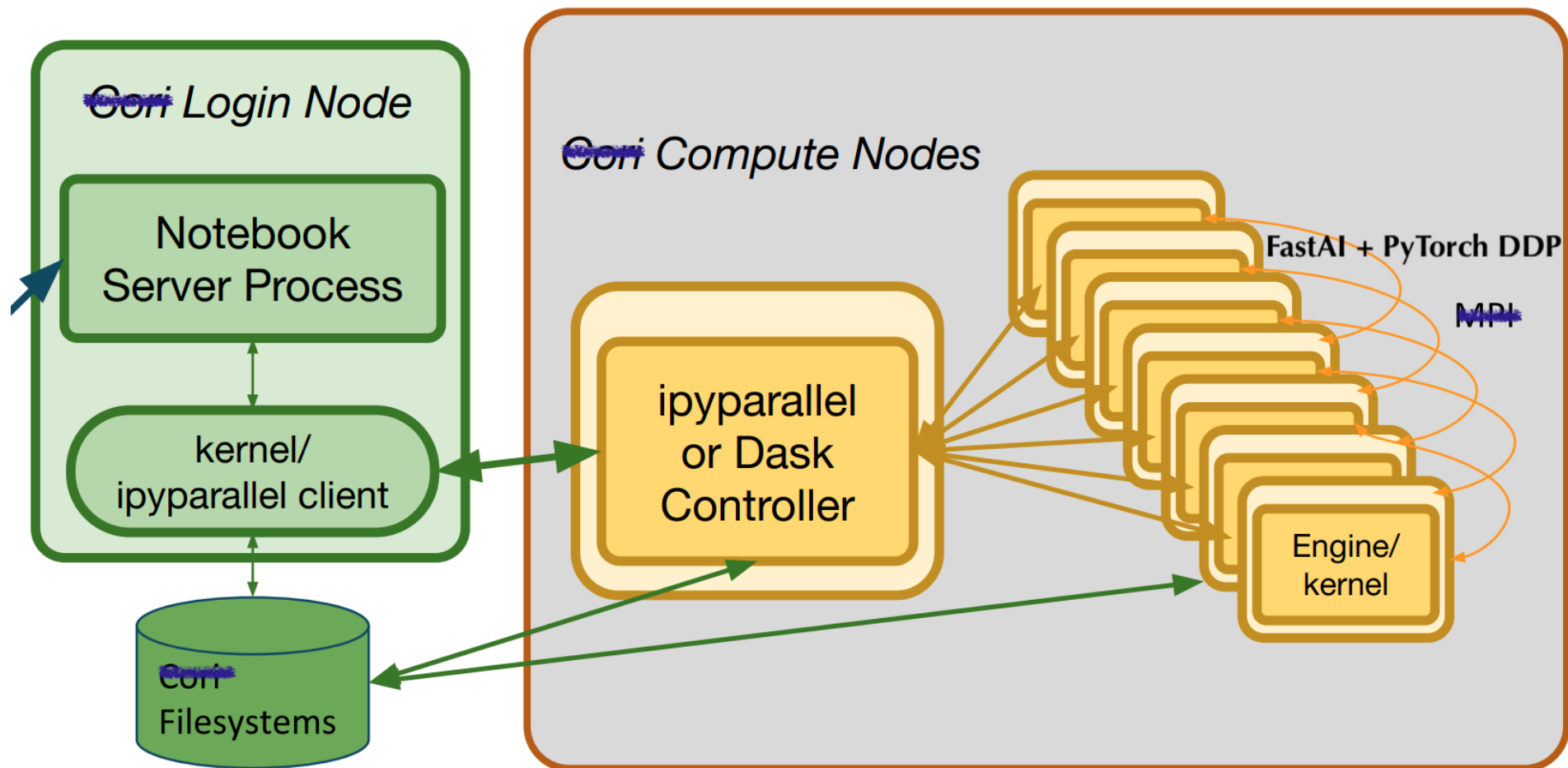
**Ddip** tries to bridge this gap.

[1] See [Reproducing DAWNBench winning results in a few lines of code](#)

[2] FastAI's tutorial on [How to launch a distributed training](#)

## **ddip** - Distributed data "interactive" parallel

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



## `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]
<a href="#">lesson3-CamVid:</a>	[3:30,4:24,12:00,12:52]	[7:33,9:12,31:50,33:40]
<a href="#">lesson3-planet:</a>	[3:20,3:45,6:15,7:30]	[4:20,5:35,14:35,18:30]
<a href="#">lesson7-superres-imagenet:</a>	[4:17]	[10:50]
lesson7-wgan:	[13:30/epoch] <b>Ouch!</b>	[4:41/epoch]

### Limitations

- Works a single host only. Luckily `ipyparallel` does support cluster of nodes.
- Not all models gain: 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](mailto:philtrade@winphil.net))

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