

BATTLE-TESTING SWARM

~~SETTING FOUNDATIONS~~

Implementing SWARM (finally)

MIKA
SENGHAAS

Next Steps

① ~~Verify 1 GPU Gist~~

NanoGPT setup
→ (GPT-2 (140m) on
Fine webEdm on
AOC)

② Implement traditional PP → Minimal (20xx)
PP Gist

③ Implement SWARM → Yandex SWARM +
Pi Zero Band

- Allow for truly distributed workers (IPv4+port, instead of local cuda worker -> might have to switch communication backend)
- Implement missing functionality (remote logging, sampling, MP)
- Benchmark speed and convergence

- Commits on Nov 15, 2024
Implement computing eval loss at step 0 mikasenghaas committed 4 days ago
Log to W&B in ckpt experiment mikasenghaas committed 4 days ago
Implement checkpointing for distributed mikasenghaas committed 4 days ago
Add pipeline benchmark (do not store intermediate tensors) mikasenghaas committed 4 days ago
Change W&B logging behaviour to use groups to group distributed runs mikasenghaas committed 4 days ago
Simplify communication mikasenghaas committed 4 days ago
Fix sampling in pipeline + overfit experiment mikasenghaas committed 4 days ago
- Commits on Nov 14, 2024
Overfit experiment in PP (WIP) mikasenghaas committed 5 days ago
- Commits on Nov 13, 2024
Minor mikasenghaas committed last week
Fix baseline training script mikasenghaas committed last week
Evaluate in debug mode mikasenghaas committed last week
Simplify configs and adjusted baseline experiments mikasenghaas committed last week
Implement custom GPT2 pytorch model mikasenghaas committed last week
Add custom GPT2 implementation mikasenghaas committed last week

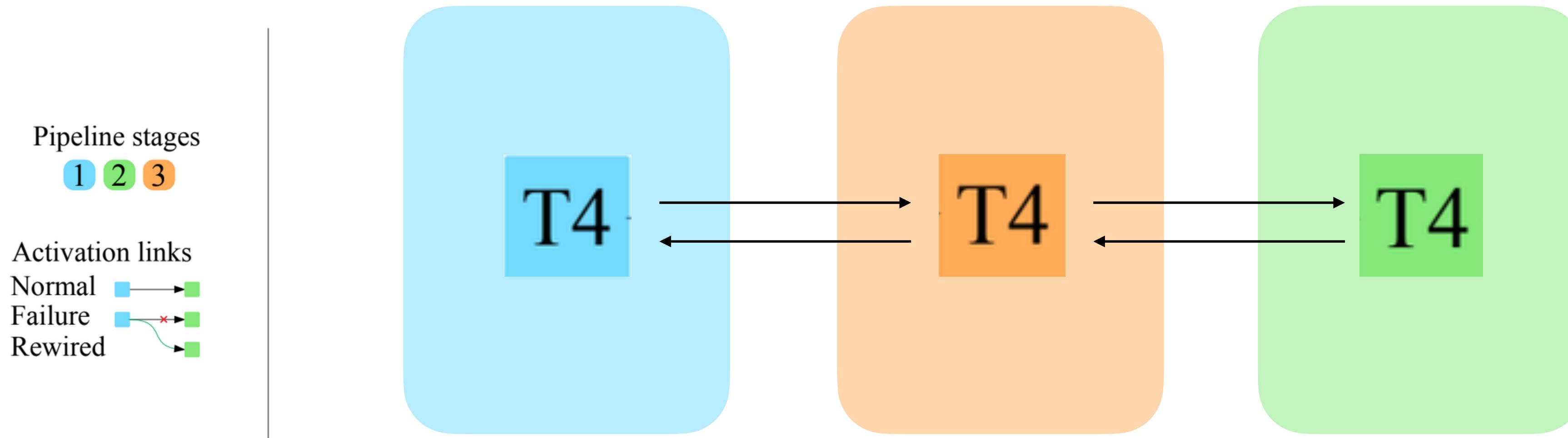
- Finished simple PP implementation
 - Sampling (needed different comm pattern)
 - Checkpointing (checkpointing local model)
 - Verification experiments (memorization + benchmarking)
 - (Also: *Removed HF dependency for model*)
 - Started SWARM implementation

Finished Implementing PP (AFAB)



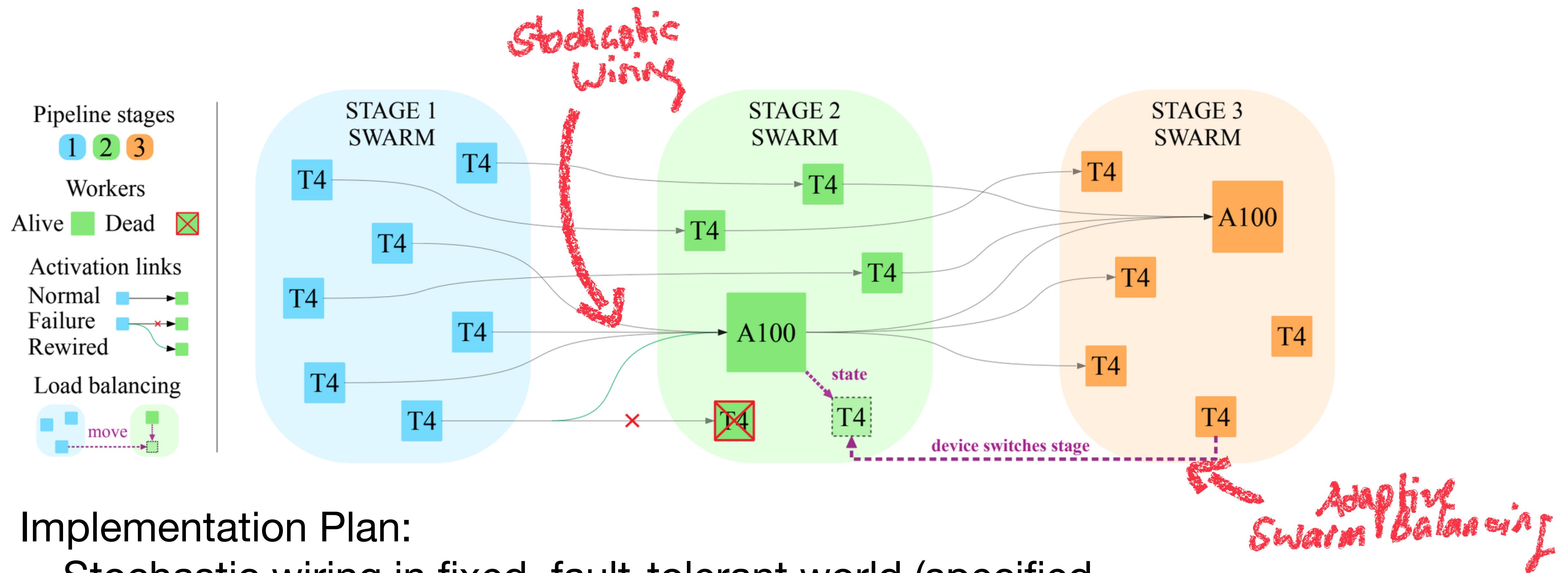
- In-/post-training evaluation
- Sampling
- Logging (Local/ W&B)
- Checkpointing
- LR Scheduling
- Mixed Precision Training

What we have



No fault tolerance !!

What we want

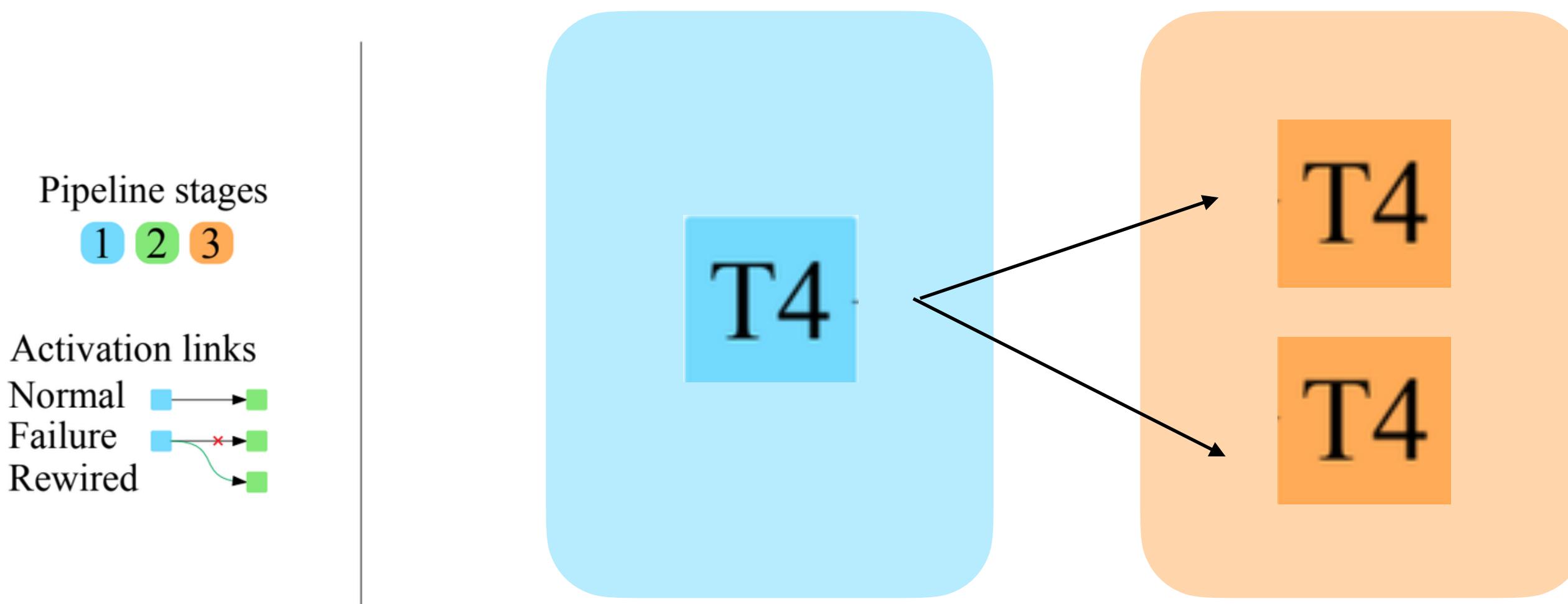


Implementation Plan:

- Stochastic wiring in fixed, fault-tolerant world (specified num_stages, ranks, world_size and uniform distribution of workers across stages -> less sync and headache)
- Fault tolerance via banning servers + periodic swarm balancing
- Move out of sandbox and to internet

Where we start: Forward pass

Send hidden states from one worker to one of two in next stage



Current problem: `torch.dist.recv()` is blocking and `torch.dist.irecv()` cannot be cancelled leading to memory errors for wrongly allocated buffers on step > 1

Idea: Pre-compute full computation graph per step? (but this is sync)

Later

- Backward pass (Backtrace computation graph to send gradients)
- Distributed data loader (Ensure that workers in first stage sample disjoint data shards, s.t. at end of step each sample is backpropagated exactly once)
- Optimize with buffers (have to be adjusted per device VRAM)
- Make work over public IP (Tailscale)
- *Dynamic resizing of world (reinitializing process group?, ...) and swarm balancing*