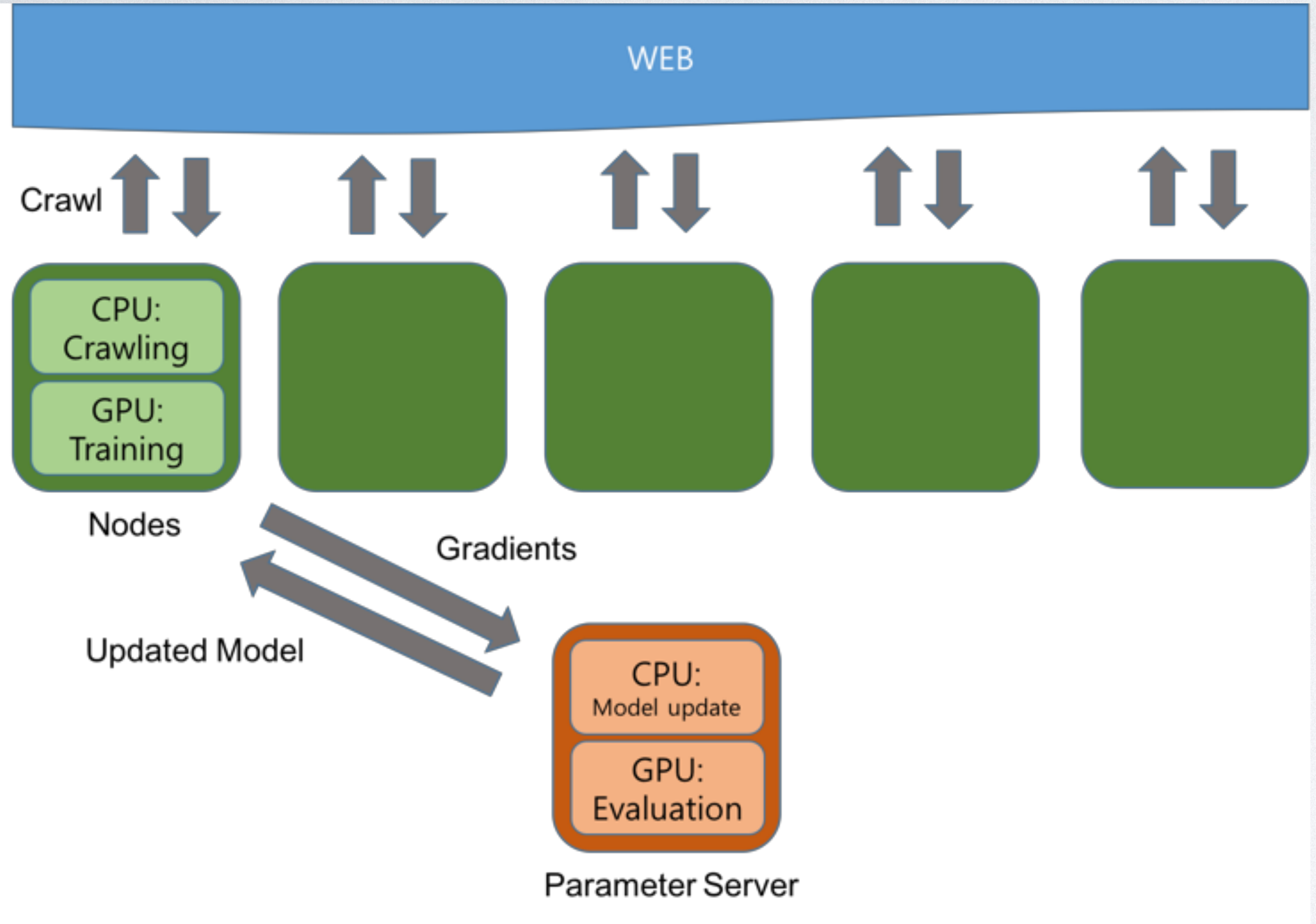
DISTRIBUTED STREAMING TEXT EMBEDDING METHOD

=> DISTRIBUTED TRAINING WITH PYTORCH

SNU 2018 - 2
BIg Data and Deep Learning
2018. 12. 18 Final Project
Team 1

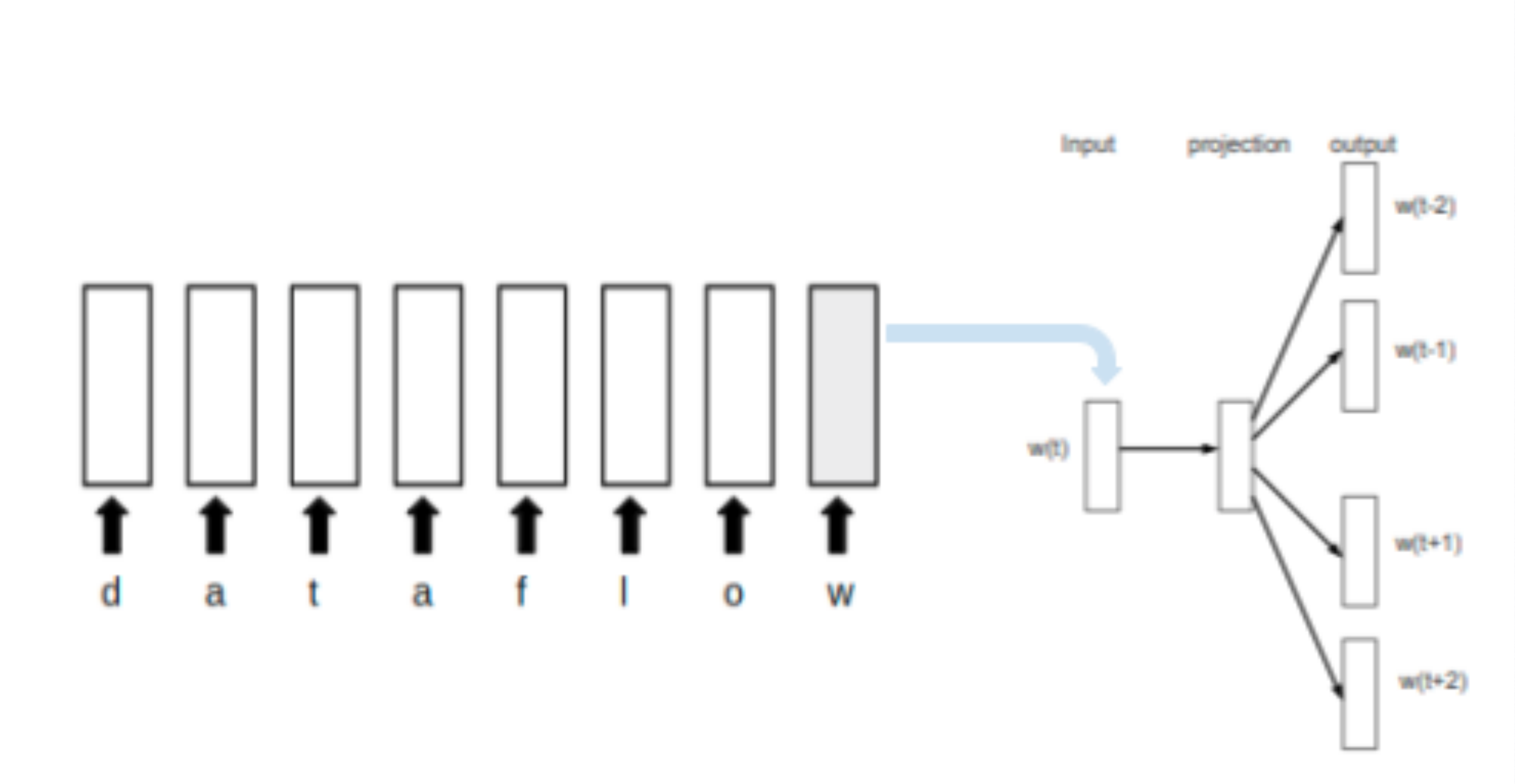
김누리, 김지영, 류성원, 이지훈

DISTRIBUTED STREAMING TEXT EMBEDDING FRAMEWORK



- Parameter Server architecture
  - Nodes
    - Crawl with CPUs
    - Train the model with GPU
  - Parameter Server
    - Model update
    - Evaluation
  - Asynchronous Update

EMBEDDING MODEL FOR STREAMING TEXT



- Character-wise word embedding with LSTM
- Skipgram Training
- Last hidden state as word embedding

PROBLEMS

1. No stable streaming datasource
2. No clear evaluation metric
3. Unstable Pytorch distributed framework

PROBLEM 1

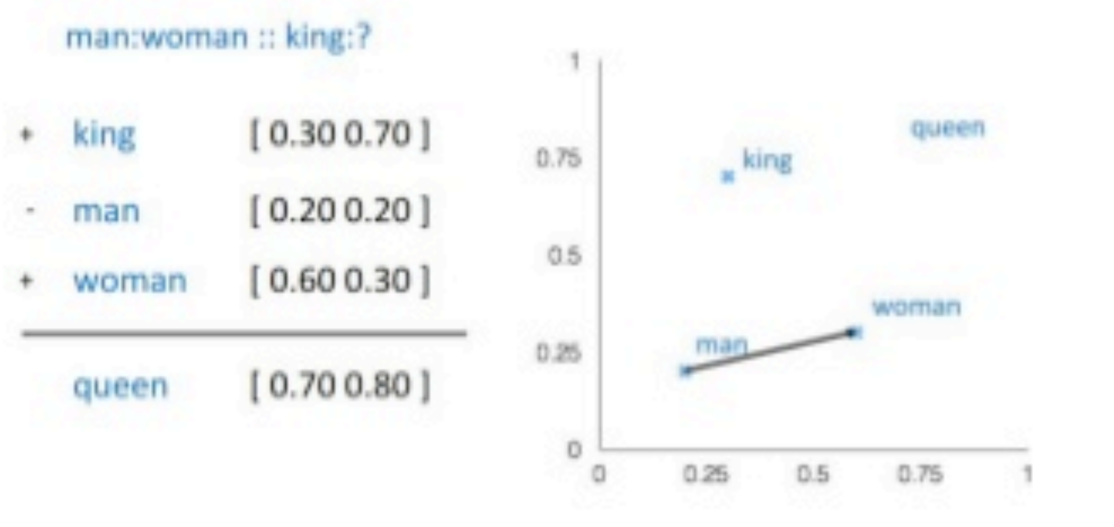
- **No stable streaming datasource**
  - Too few machines
  - Crawling APIs are extremely unstable (Facebook, Youtube, Twitter)
  - Crawling bottleneck >> GPU bottleneck
  - => Check validity of distributed word embedding and our model

PROBLEM 2

Word 1	Word 2	Human (mean)
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	internet	7.58
plane	car	5.77
professor	doctor	6.62
stock	phone	1.62
stock	CD	1.31
stock	jaguar	0.92

- **No clear evaluation metric**
  - Word similarity task
    - MEN, MTurk, RW, SimLex999, WS353
  - Word analogy task
    - Google analogy, MSR analogy
  - Need to train with dataset that contains all the words
    - Wikipedia dataset: 32GB text, 320GB when preprocessed
    - **Takes Forever**

$$x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$$



PROBLEM 2

- Solution: **PIP Loss***
  - Metric to measure distance between embeddings
  - Exploit unitary invariance property of embeddings
  - $\mathrm{PIP}(E) = EE^T \quad \|\mathrm{PIP}(E_1) - \mathrm{PIP}(E_2)\| = \|E_1E_1^T - E_2E_2^T\| = \sqrt{\sum_{i,j}(\langle v_i^{(1)}, v_j^{(1)}\rangle - \langle v_i^{(2)}, v_j^{(2)}\rangle)^2}$

- The Ground truth of Skip-gram: SPPMI matrix*
  - $\mathrm{PMI}_{ij} = \log \frac{p(v_i, v_j)}{p(v_i)p(v_j)}$
  - PIP Loss with SPPMI matrix can be used as evaluation metric

Source: Yin, Zi, and Yuanyuan Shen. "On the dimensionality of word embedding." *Advances in Neural Information Processing Systems*. 2018.
Levy, Omer, and Yoav Goldberg. "Neural word embedding as implicit matrix factorization." *Advances in neural information processing systems*. 2014.

PROBLEM 3

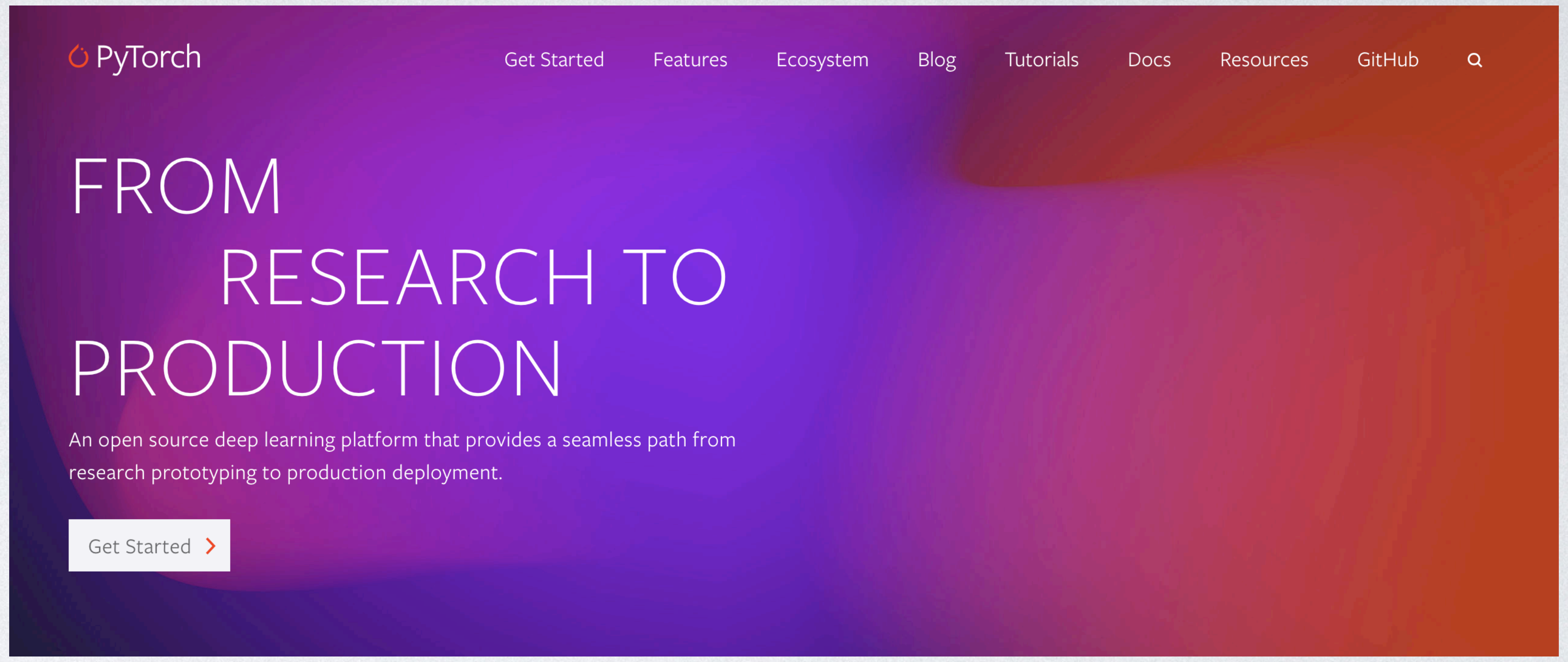
- **Unstable Pytorch distributed framework**
    - Data parallel

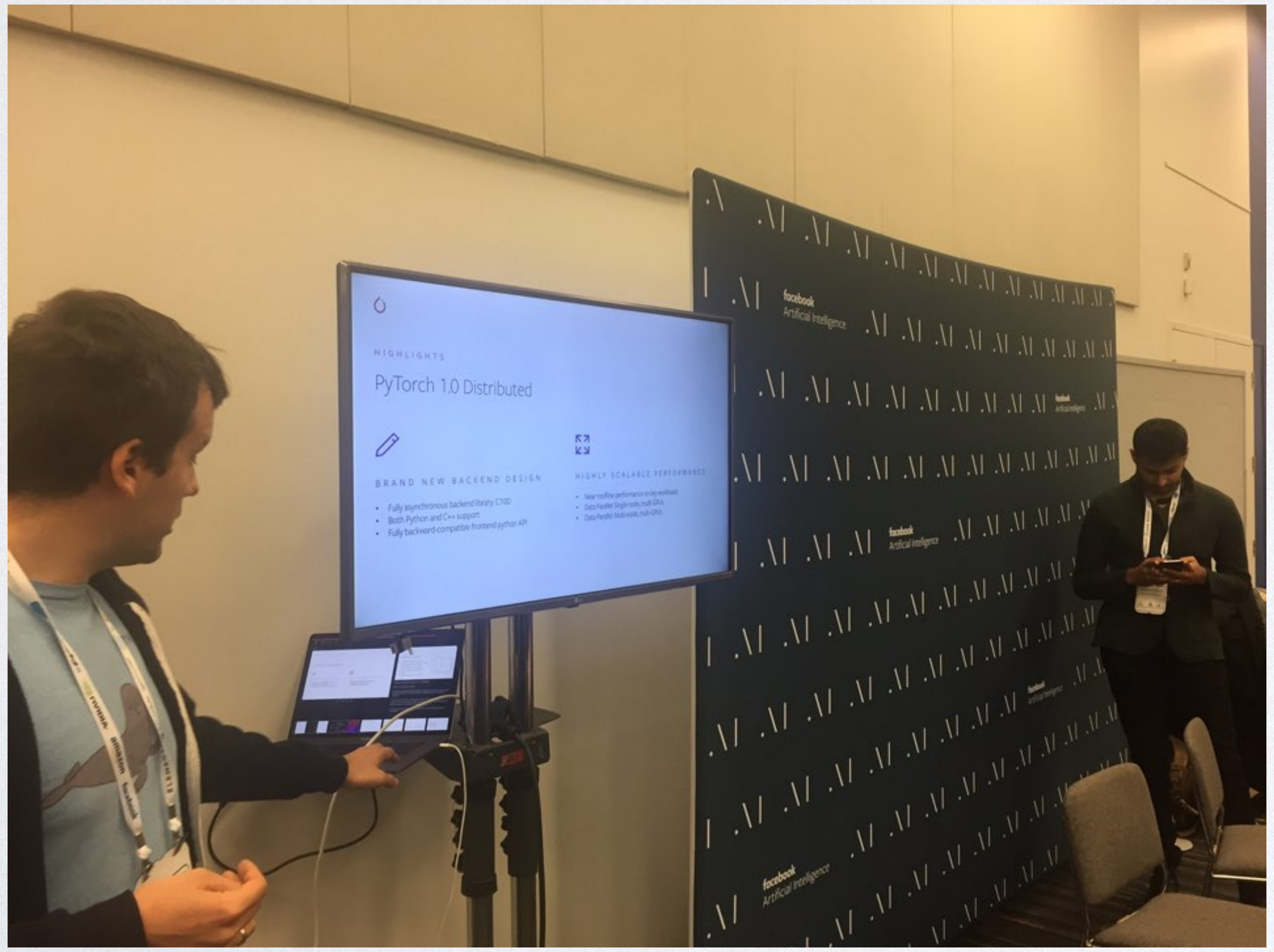
```
model = Model(input_size, output_size)
if torch.cuda.device_count() > 1:
  print("Let's use", torch.cuda.device_count(), "GPUs!")
  # dim = 0 [30, xxx] -> [10, ...], [10, ...], [10, ...] on 3 GPUs
  model = nn.DataParallel(model)

model.to(device)
```

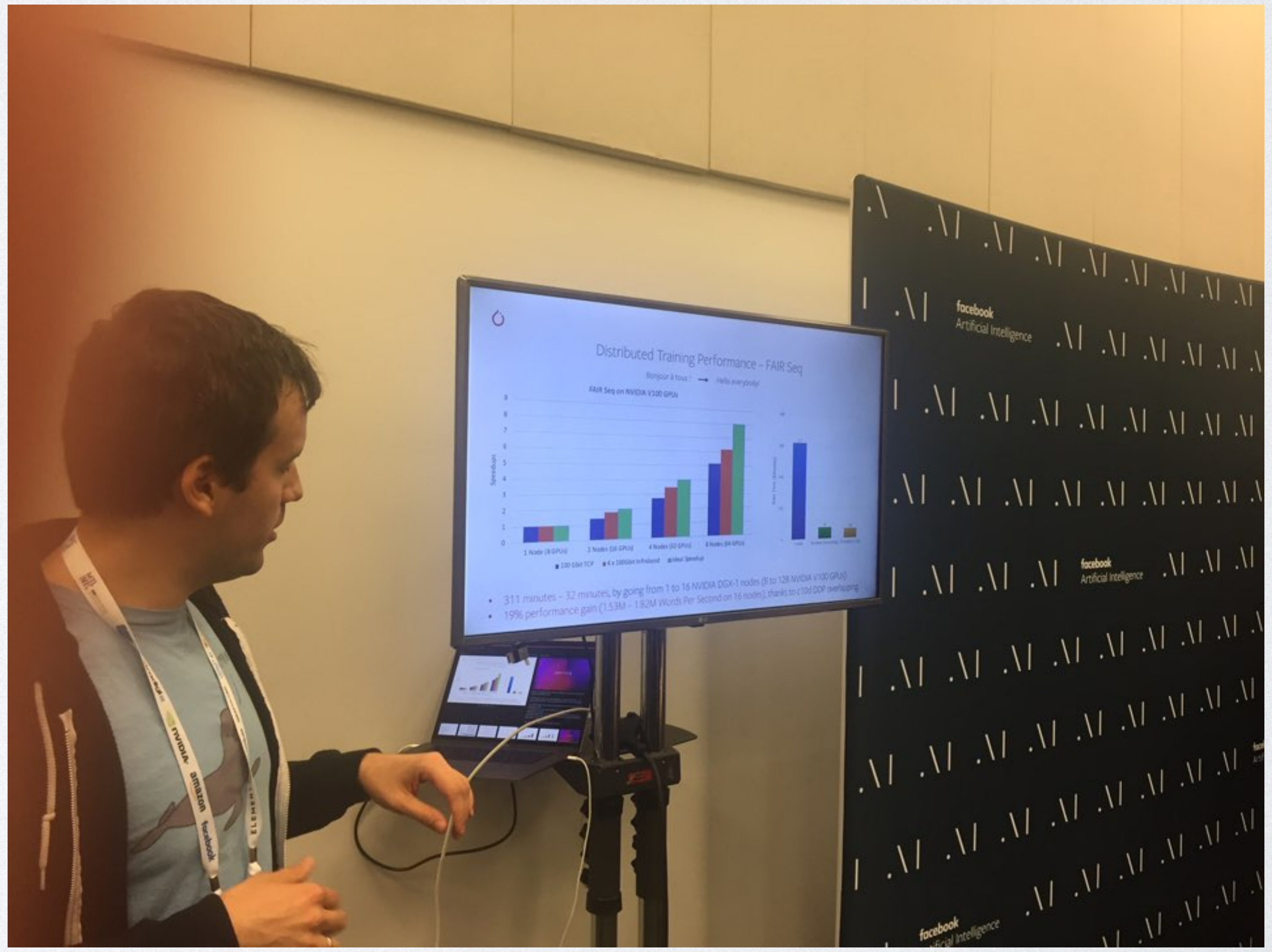
```
def data_parallel(module, inputs, device_ids=None, output_device=None, dim=0, module_kwargs=None):
    if not isinstance(inputs, tuple):
        inputs = (inputs,)

    if device_ids is None:
        device_ids = list(range(torch.cuda.device_count()))

    if output_device is None:
        output_device = device_ids[0]

    inputs, module_kwargs = scatter_kwargs(inputs, module_kwargs, device_ids, dim)
    if len(device_ids) == 1:
        return module(*inputs[0], **module_kwargs[0])
    used_device_ids = device_ids[:len(inputs)]
    replicas = replicate(module, used_device_ids)
    outputs = parallel_apply(replicas, inputs, module_kwargs, used_device_ids)
    return gather(outputs, output_device, dim)
```

PROBLEM 3

- **Pytorch 1.0**
  - Distributed Library
    - Synchronous
    - Asynchronous







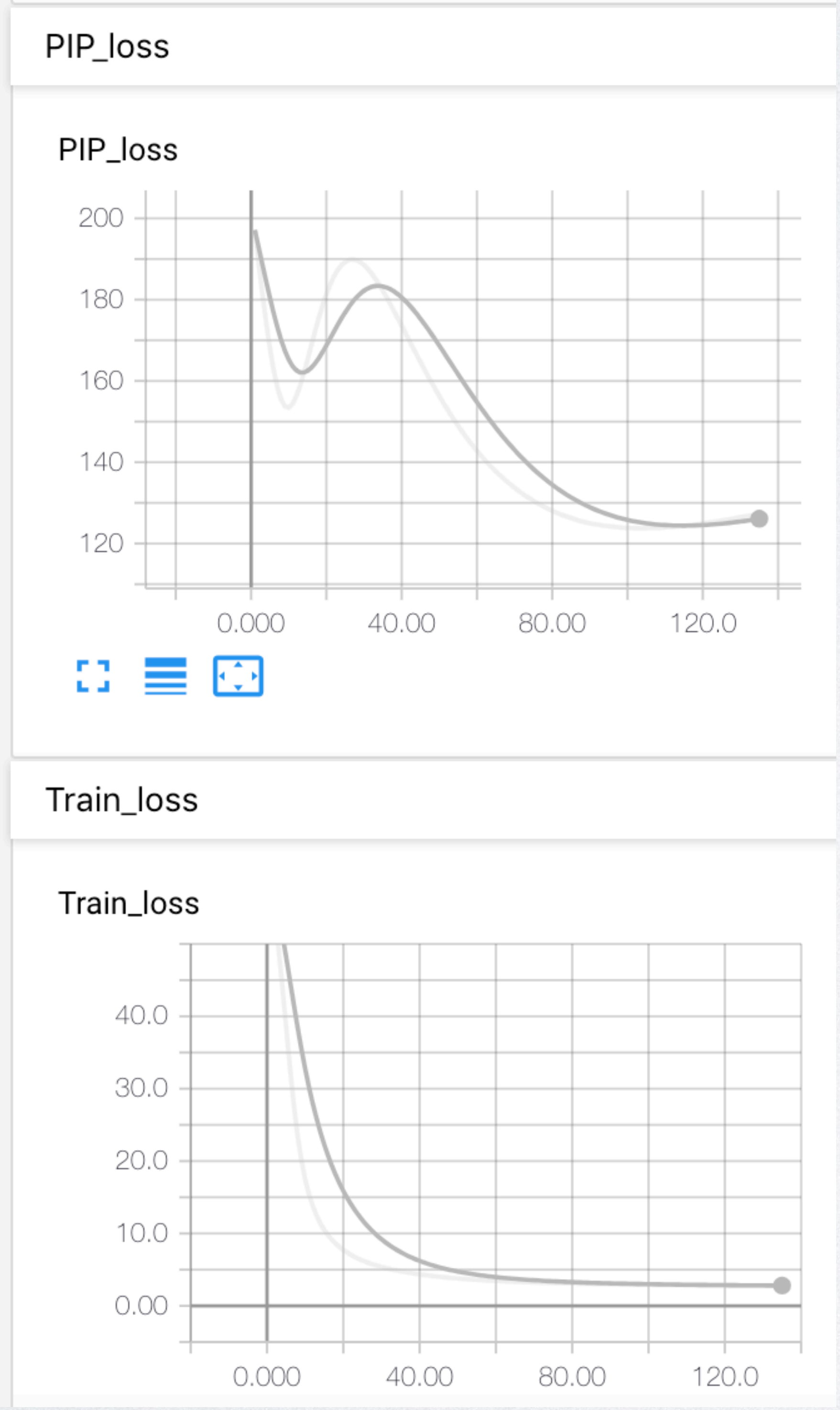
EXPERIMENT SETUP

- SGNS
  - 6Mb text dataset
    - Harry Potter Series
  - Tokenized / lemmatized
  - window: 5 / ns: 10 / threshold: 3 / subsample: 2e-3
  - Learning Rate: 1e-4
  - epoch: 300

- Pytorch
  - 1 process no GPU
  - 1 process one GPU (970)
  - 1 process 4 GPUs (970)
  - 4 process 4 GPUs (Ethernet)
    - Asynchronous
    - Synchronous

EXPERIMENT RESULT 1

- Embedding size: 200
- Batch size: 1024



	Average time per epoch	Throughput	Best PIP loss
1 process 1 GPU	34.10	98,212.7	123.6
1 process 4 GPUs	25.37	132,060.5	129.6
Cluster	**394.27**	8,494.3	?

Source: "Distributed Streaming Text Embedding Method", Sungwon Lyu, Jeeyung Kim, Noori Kim, Jihoon Lee, Sungzoon Cho, Korea Data Mining Society 2018 Fall Conference, Special Session

EXPERIMENT RESULT 2

- Embedding size: 200
- Batch size: 8192

	Average time per epoch	Throughput	Best PIP loss
1 process 1 GPU	28.6	117,099.8	129.3
1 process 4 GPUs	24.1	138,964.9	-
Cluster (Sync)	52.79	63,441	193.6
Cluster (Async)	46.5	72,022.6	?

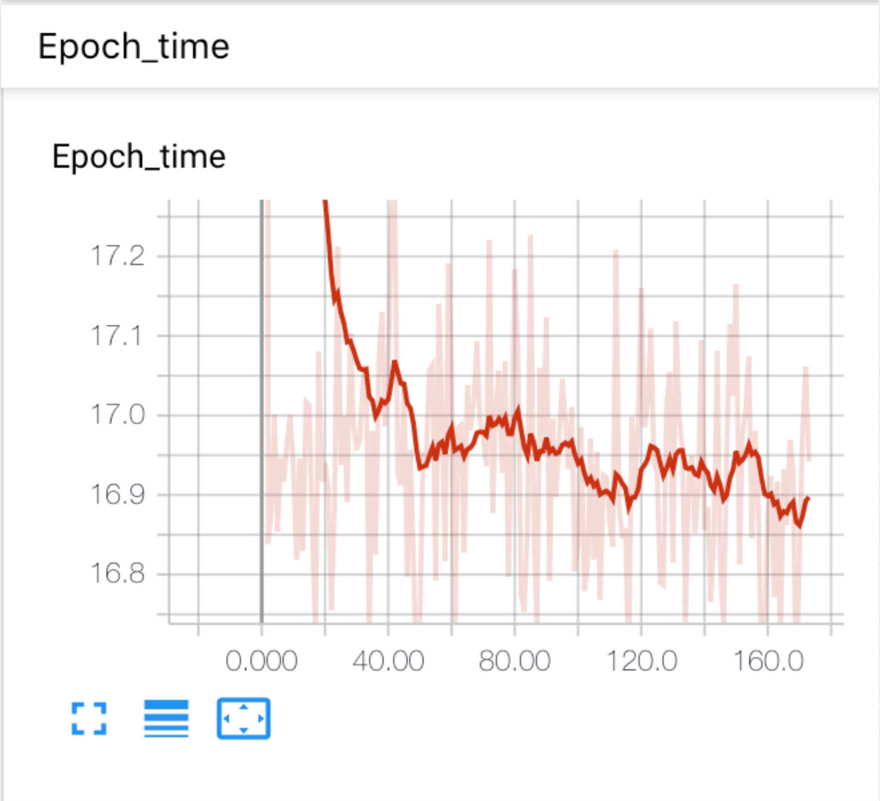
EXPERIMENT RESULT 3

- Embedding size: 50
- Batch size: 1024

	Average time per epoch	Throughput	Best PIP loss
1 process 1 GPU	21.6	155,048.8	14.52
1 process 4 GPUs	24.08	139,080.3	15.44
Cluster	93.81	35,700.4	44.21

EXPERIMENT RESULT 4

- Embedding size: 50
- Batch size: 8192



	Average time per epoch	Throughput	Best PIP loss
1 process 1 GPU	29.32	114,224.2	15.19
1 process 4 GPUs	21.28	157,380.3	-
Cluster	**16.93**	197,817.7	44.12

RESULT SUMMARY

model	node	sync	gpu	embedding	batch	time/epoch	lowest PIP loss
sgns	4	async	4	200	8192 * 4	46.5	X
sgns	4	sync	4	200	8192 * 4	52.79	193.6
sgns	4	sync	4	200	1024 * 4	394	X
sgns	4	sync	4	50	8192 * 4	16.93	44.12
sgns	4	sync	4	50	1024 * 4	93.81	44.21
sgns	1	-	1	200	8192	28.6	129.3
sgns	1	-	1	200	1024	34.1	123.6
sgns	1	-	1	50	8192	29	15.1885
sgns	1	-	1	50	1024	21.6	14.52
sgns	1	-	4	200	8192 * 4	24.1	ing
sgns	1	-	4	200	1024 * 4	25.37	129.6
sgns	1	-	4	50	8192 * 4	21.28	ing
sgns	1	-	4	50	1024 * 4	24.08	15.44
rnn	1	-	1	200	1024	1133.9	1.11

CONCLUSION

- Single node is usually better when cluster is not big enough
- **Less communication (more batch size, less weights) leads to faster training**
- The quality of the word embedding is affected by batch-size (smaller seems better)
- Therefore, sparse word embedding is not appropriate for distributed training

FUTURE WORK

- Do experiment with dense model
- Compare with Tensorflow / with PS architecture
- Try Ring all-reduce
- Find way to minimize the communication