SciPy India
29&30 November 2019

Daskify an MPI application for distribution using Dask

Learnings during Implementation

IBM Extreme Blue Interns (NITK)
Melchizedek Das
Paras Bhagtya

IBM
Sangeeth Keeriyadath
Pradipta Ghosh
Sivakumar Krishnasamy

Agenda

- Motivation
- Goal
- Dask primer
- Dask Distribution methods used
- Substitutes for MPI reduction operations
- The path ahead

Where is time spent in Machine Learning?

- **Time Spent with Machines**

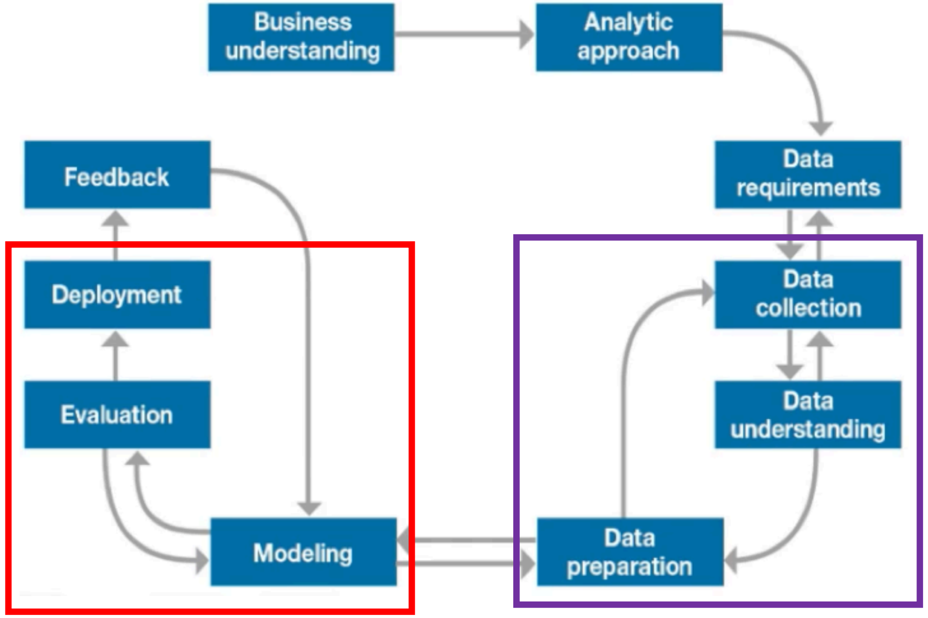
  Number crunching / Finding the best hyper parameters

  "Prototyping and experimenting with machine learning were mentioned by more than half of respondents."

- **Time Spent by Data Scientists**

  Feature Engineering

  " ..over 75% suggested understanding and analyzing the data is a common activity."



https://courses.cognitiveclass.ai/

(Based on Kaggle survey 2019 - https://www.kaggle.com/kaggle-survey-2019)

Reduce time spent with machines

- **Hardware design (e.g. IBM Power System AC922)**

Summit and Sierra remain in the top two spots.
IBM-built supercomputers employing Power9 CPUs and NVIDIA Tesla V100 GPUs.

- **ML library utilize advances in hardware and algorithms (e.g. Snap ML)**
  - Scale out “Distributed training” implementation for massive datasets (Supports MPI and Spark)
  - Specialized solvers designed for “GPU acceleration”
  - Optimized algorithms for “Sparse data structures”

Ref - https://www.top500.org/lists/2019/11/ , https://www.zurich.ibm.com/snapml/

Reduce time spent by Data Scientists

Building Blocks in Python ecosystem -

- NumPy (Fundamental package for scientific computing)
- Pandas (Fast and flexible data analysis library)

Handle Big Data in Pythonland. And DASK was born!

- Dask Array (scales Numpy)
- Dask DataFrame (scales Pandas)

Learn more - https://dask.org/

Goal

Use Dask distributed processing for data exploration and feature engineering

feed ↓ into

State-of-the-art distributed machine learning library SnapML (pai4sk package) for training

"JupyterLab and its offshoots are the most common, with 83% of data scientists using it on a regular basis." (https://www.kaggle.com/kaggle-survey-2019)

Agenda

- Motivation
- Goal
- Dask primer
- Dask Distribution methods used
- Substitutes for MPI reduction operations
- The path ahead

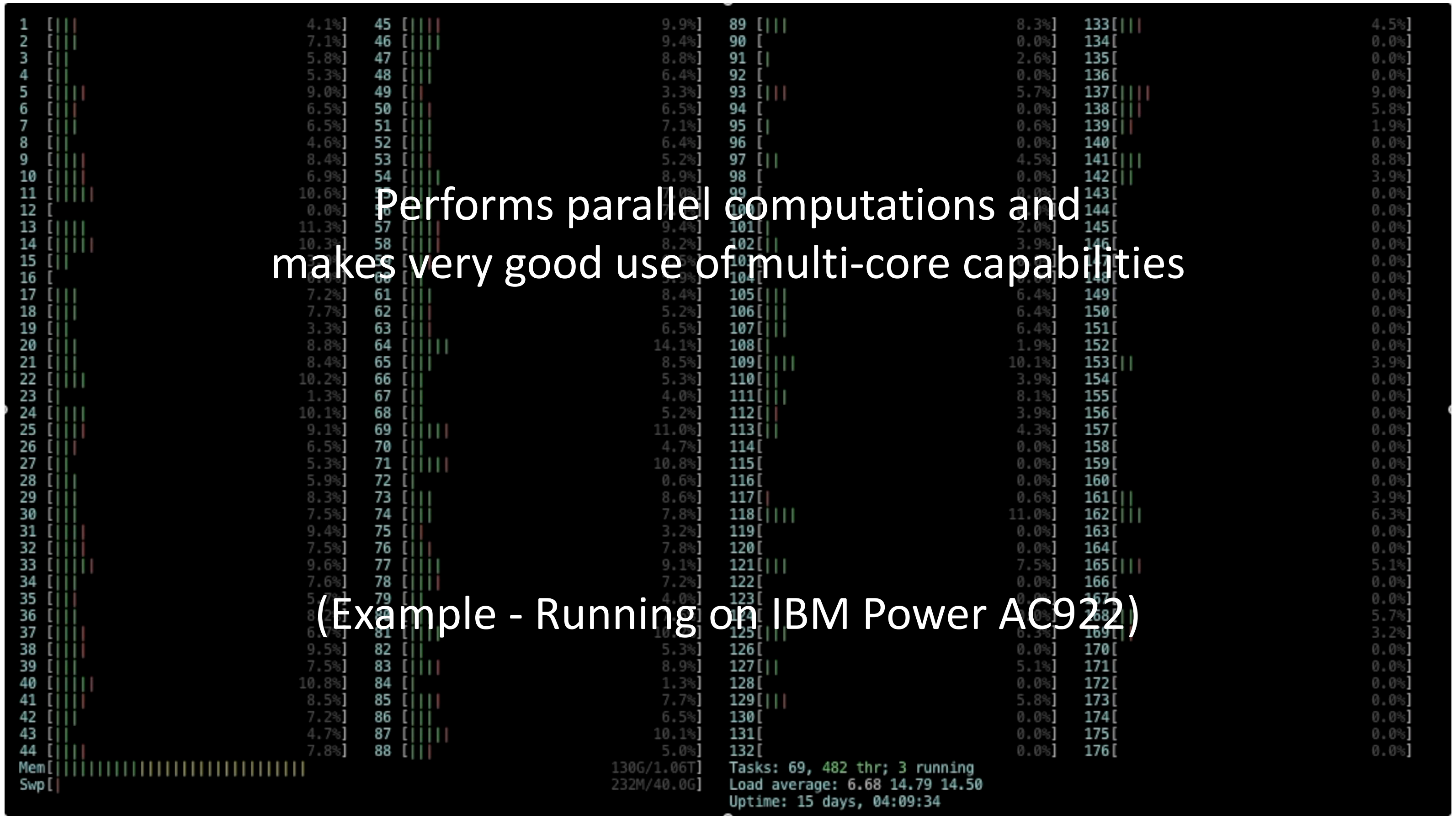
Dask setup

- A distributed dask cluster can be set up in multiple ways and offers more features
- Dask Scheduler
- Dask Worker
- Dask Dashboard
- Dask Custom Configuration

Learn more - https://docs.dask.org/en/latest/setup.html

Dask DataFrame / Array

- Work like Pandas and Numpy, but at scale



Dask Client 101

- Initialize a client by pointing to address of the dask scheduler
```
client = Client('9.3.89.44:8786’)
```

- Runs all dask collections (dataframe, array etc.) in the distributed cluster
```
client.who_has(X_da_train)
{"('array-191b9b48149b501ba630b8426a65fd6e', 1, 0)":
('tcp://9.3.89.44:40229',),
"('array-191b9b48149b501ba630b8426a65fd6e', 2, 0)":
('tcp://9.3.89.27:36945’,),}
```

- Submit a function to the scheduler
```
future = client.submit(get_unique_labs, data)
```

- Wait until computation completes, gather result to local process.
```
future.result()
```

Learn more - https://distributed.dask.org/en/latest/client.html

Substitute for MPI_Allreduce(.. MPI_SUM ..)

Get total label count for each class in the entire dataset

```
uint32_t num_pos = data->get_num_pos();
uint32_t num_neg = data->get_num_neg();
MPI_Allreduce(MPI_IN_PLACE, &num_pos, 1, MPI_UNSIGNED, MPI_SUM, MPI_COMM_WORLD);
MPI_Allreduce(MPI_IN_PLACE, &num_neg, 1, MPI_UNSIGNED, MPI_SUM, MPI_COMM_WORLD);
```

```
num_pos = da.sum(da.array(y) > 0 ).compute()
num_neg = total_ex - num_pos
```

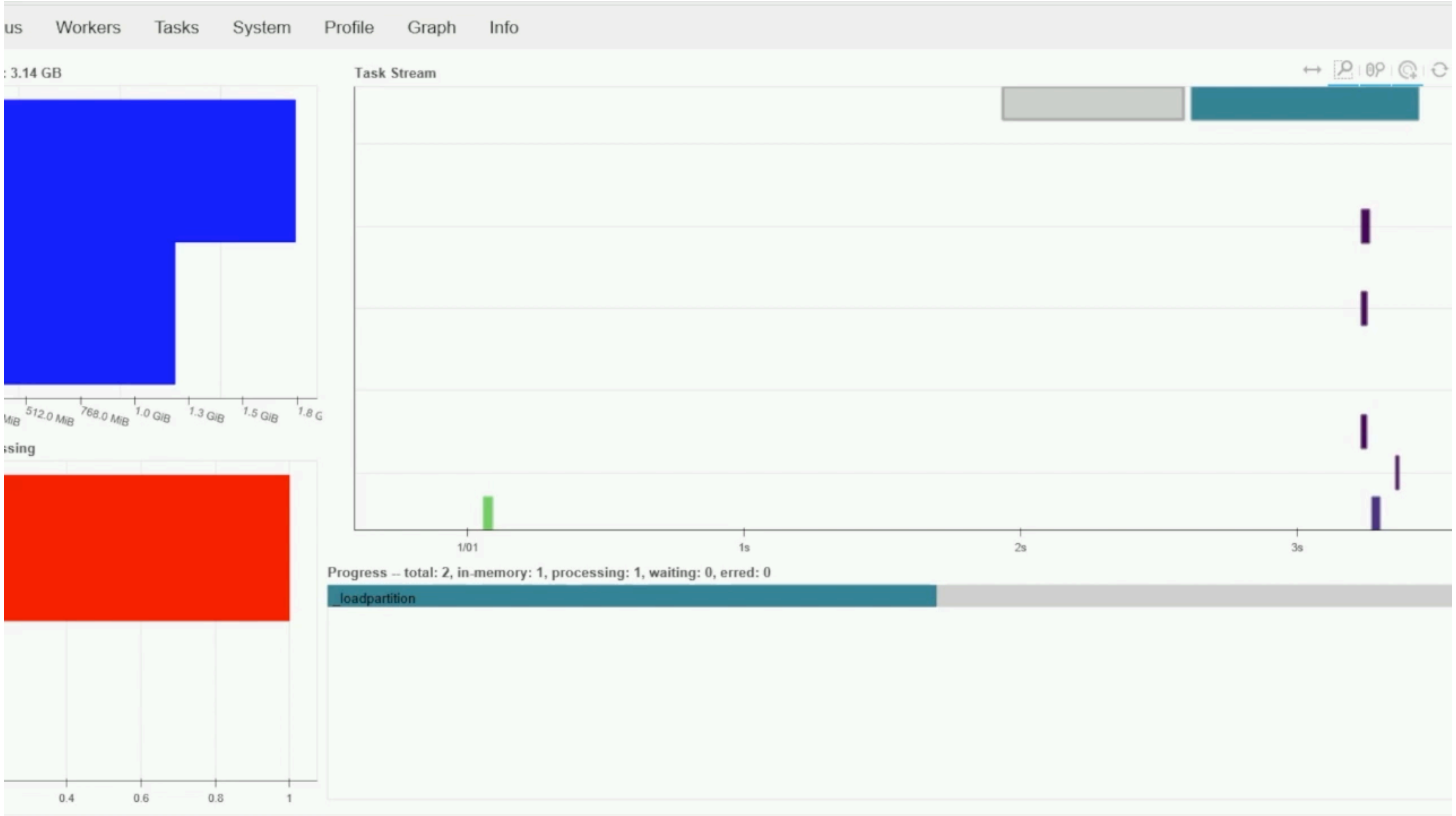
Substitute for MPI_Allreduce(.. MPI_LOR ..)

Have the workers converged for their partitions

```
MPI_Allreduce(MPI_IN_PLACE, stop_partition, num_partitions, MPI_INT, MPI_LOR,
MPI_COMM_WORLD);

stop = true;

for (uint32_t i = 0; i < num_partitions; i++) {

    stop &= stop_partition[i];

}
```

```
stop=1

for i in range(len(stop_partition)):
   stop=stop & stop_partition [i]
```

How did the Dashboard feel about SnapML?



The path ahead

- Performance consideration
  - Overhead of switching from C++ library to Python for MPI substitute reduction operations
  - Use dask-cuda for GPU solvers (Improve deployment and management of Dask workers on CUDA-enabled systems)
  - Rechunk ahead of time or not?

- Observations
  - With bigger datasets, needed to restart the distributed network (`client.restart()`) after each job. Memory leaks?
  - Sparse Arrays in Dask are sparse.COO format. Needs convertion to scipy.sparse.csr_matrix/csc_matrix for fast arithmetic operations

Recap

- Motivation
- Goal
- Dask primer
- Dask Distribution methods used
- Substitutes for MPI reduction operations
- The path ahead

We are a big family! Even bigger ☺

IBM Zurich Research Lab

Andreea Anghel

Dimitrios Sarigiannis

Haris Pozidis

Nikolas Ioannou

Thomas Parnell

IBM India DML squad

Josiah Samuel

Pavani Vemuri

Poornima Nayak

Pradipta Ghosh

Ravi Gummadi

Sangeeth Keeriyadath

Sivakumar Krishnasamy

धन्यवाद / Thank you

Sangeeth Keeriyadath (k.sangeeth@in.ibm.com)

Pradipta Ghosh (pradghos@in.ibm.com)

Sivakumar Krishnasamy (sikrishn@in.ibm.com)

Extras…

Considerations to feed Dask Array to SnapML

- Use one chunk of dask array per node (use `X.rechunk()`)
- Compute the globally required data using dask array primitives –

```
total_num_examples = X.shape[0]
num_positive_labels = da.sum(y > 0 ).compute()
```

- Extract numpy array from data (use `X.compute()`), and send it to C++ library

Submitting work to Dask Cluster

- Want the processing to start immediately, but don't wait for the result

```
futures=[]
for i in range(len(data)):
    futures.append(client.submit(get_unique_labs, data[i], workers=worker[i]))
```

client.map() ?

- Wait for the results only when required

```
local_unique_labs_dict_list=[]
for i in range(len(futures)):
    local_unique_labs_dict_list.append(futures[i].result())
```

client.gather() ?

Learn more - https://docs.dask.org/en/latest/futures.html

Substitute for MPI_Allreduce(.. MPI_MAX ..)

Validate if any data partition for binary classification has different labelling method e.g. {-1, 1} and {0, 1}

```
MPI_Allreduce(MPI_IN_PLACE, &is_zero, 1, MPI_UNSIGNED, MPI_MAX, MPI_COMM_WORLD);
MPI_Allreduce(MPI_IN_PLACE, &is_one, 1, MPI_UNSIGNED, MPI_MAX, MPI_COMM_WORLD);
MPI_Allreduce(MPI_IN_PLACE, &is_minus_one, 1, MPI_UNSIGNED, MPI_MAX, MPI_COMM_WORLD);
```

```
for x in unique_labs:
    if x == 0:
        is_zero=True
    if x == 1:
        is_one=True
    if x == -1:
        is_minus_one=True
```

MPI_Send / MPI_Recv

```
Loop through and send to each remote node..
       MPI_Send(&local_num_ulabs, 1, MPI_INT, node, …);
       MPI_Send(buf, count, MPI_FLOAT, node, …);
Loop through and receive from each remote node..
       MPI_Recv(&remote_num_ulabs, 1, MPI_INT, …);
       remote_unique_labs.resize(remote_num_unique_labs, 0);
       MPI_Recv(&remote_unique_labs[0], remote_num_unique_labs, MPI_FLOAT, …);
```

```
for i in range(len(data)):
    worker_futures.append(client.submit(get_unique_labs, data[i], workers=worker[i])

local_unique_labs_dict_list=[]
for i in range(len(worker_futures)):
    local_unique_labs_dict_list.append( worker_futures[i].result() )

unique_labs_dict = dict(functools.reduce(operator.add,
                        map(collections.Counter, local_unique_labs_dict_list)))
```