Seastar

Release 1.0.0

Joel Mathew Cherian, Nithin Puthalath Manoj

Nov 03, 2022

CODEBASE

1 What is Seastar? 3

2 Indices and tables 7

Index 9

This documentation was made to understand how the Seastar compiler works. Feel free to use it for your project and learning purposes.

CHAPTER

ONE

WHAT IS SEASTAR?

Seastar is a system for programming GNN models using a vertex-centric approach. This new programming approach ensures that the logic for various GNN models can be written intuitively. It also allows those reading the code to ascertain the models purpose. The Seastar system can be used to implement models like GCN and GAT with relative ease.

While Seastar provides a new programming approach, it can perform optimizations on the model to bridge some of the gaps in other GNN frameworks like DGL and PyG. Seastar takes a function representing the operations performed on a single vertex as input.

Seastar is successful in achieving lesser memory consumption and faster execution in comparison to PyG and DGL.

1.1 exp

`exp` (experiments) directory contains code for different GNN models like `gcn`, `gat`, `rgcn` and `appnp` with which the datasets are trained on. Each of the model stated above is implemented using `dgl`, `pyg` and `seastar`.

The `result` directory has a csv file *final_result.csv* which contains the results of the experiment/training conducted by each GNN model on the datasets.

1.1.1 APPNP

1.1.2 GAT

1.1.3 GCN

1.1.4 RGCN

1.1.5 run_exp

This is where the entire program starts from, when you run it on Google Colab or any other platform.

Note: This is not where Seastar starts from

create_exp_list_sample0()

create_exp_list_sample0(*args*)

According to the different GNN models passed as input through the command line, create_exp_list_sample0 returns a list (*exp_list*) of GNNExp class objects.

Parameters

args – Contains the arguments passed through the command line. Arguments are models, systems, gpu and num_epochs. eg: args = Namespace(gpu=0, models=['gcn'], num_epochs=200, systems=['dgl', 'seastar'])

Return type

List of GNNExp class objects. These objects can be from the following classes: GATExp, GCNExp, APPNPExp and RGCNExp

main()

main(*args*)

Format the exception with a traceback.

Parameters

args – Contains the arguments passed through the command line. Arguments are models, systems, gpu and num_epochs. eg: args = Namespace(gpu=0, models=['gcn'], num_epochs=200, systems=['dgl', 'seastar'])

Return type

None

if __name__ == __main__:

```
parser = argparse.ArgumentParser(description='Experiment')
```

Within this if-block, the arguments which are passed through the command line are parsed and stored inside an *argparse.ArgumentParser* object.

```
parser.add_argument("--models", nargs='+', default='gat', help="which models to run.␣
↪Example usage: --models gat gcn")
parser.add_argument("--systems", nargs='+', default='dgl', help="which models to run.␣
↪Example usage: --systems dgl seastar pyg")
parser.add_argument("--gpu", type=int, default=0, help="which GPU to use. Set -1 to use␣
↪CPU.")
parser.add_argument("--num_epochs", type=int, default=200, help="number of training␣
↪epochs")
```

- –models: Refer to the GNN models which are to be used for training eg: GCN, GAT, etc.
- –systems: Refer to the frameworks which are used for implementing these models eg: DGL, PyG and Seastar
- –gpu: To whether use a gpu or not
- –num_epochs: The number of epochs required for training

```
args = parser.parse_args()
args.models = [args.models] if not isinstance(args.models, list) else args.models
args.systems = [args.systems] if not isinstance(args.systems, list) else args.systems
```

Parses the input command and stores the arguments in *args* . Also makes sure that *args.models* and *args.systems* are list datatype.

This is what args would look like for the following command line input:

```
>> ./run_exp.py --models gcn --systems dgl seastar
args = Namespace(gpu=0, models=['gcn'], num_epochs=200, systems=['dgl', 'seastar'])
```

Attention: Need to figure out `nb_access` and `dataset` directory

1.2 python

1.2.1 seastar

Compiler Interface (zoom_v2)

class Context

GraphInfo is a named tuple data structure whose fields can be accessed using indexing and the following field names

`GraphInfo(number_of_nodes, number_of_edges, in_row_offsets, in_col_indices, in_eids, out_row_offsets, out_col_indices, out_eids, nbits)`

_f

_nspace

_entry_count: int

_run_cb

_input_cache: dict

_graph_info_cache

_executor_cache

class CtxManager

_ctx_map: dict

_run_cb

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

_ctx_map (*CtxManager attribute*), 5
_entry_count (*Context attribute*), 5
_executor_cache (*Context attribute*), 5
_f (*Context attribute*), 5
_graph_info_cache (*Context attribute*), 5
_input_cache (*Context attribute*), 5
_nspace (*Context attribute*), 5
_run_cb (*Context attribute*), 5
_run_cb (*CtxManager attribute*), 5

B

built-in function
    create_exp_list_sample0(), 4
    main(), 4

C

Context (*built-in class*), 5
create_exp_list_sample0()
    built-in function, 4
CtxManager (*built-in class*), 5

M

main()
    built-in function, 4