CSE530: Assignment#1 Report

Ajay Narayanan Sridhar , PSU ID : 970750943

In this notebook, we will be performing ablation studies on the effect of cache parameters on the following kernels

- 2-D Matrix column wise copy
- 2-D Matrix transpose
- 2-D Matrix gather
- 2-D Matrix scatter

NOTE:

- Kernel code can be found in ./content/CachePerformanceOnMatMul/src
- Traces can be found in ./content/CachePerformanceOnMatMul/traces
- Logs can be found in ./content/CachePerformanceOnMatMul/logs
- All our experiments use a matrix size of 10x10 with a sparsity of 50% and has no optimization on storing sparse matrices.

In [ ]:
```
import pathlib
import pickle
import pprint
import matplotlib.pyplot as plt
```

In [ ]:
```
# setup root directory path
abs_root_dir = pathlib.Path().parent.resolve()
print(abs_root_dir)
```

```
/home/ajay/Documents/CSE_530/Assignments/assignment_1/question
```

In [ ]:
```
# change directory to the source files
%cd $abs_root_dir/content/CachePerformanceOnMatMul/
```

```
/home/ajay/Documents/CSE_530/Assignments/assignment_1/question/content/Cach
ePerformanceOnMatMul
```

In [ ]:
```
# To run experiments again, use the following command
!source ./run_kernel.sh 10 10 1 '50' input_matrix.in true clean
```

In [ ]:
```
traces = ["mat_scatter", "mat_gather", "mat_transpose", "mat_column_wise_cop
```

In [ ]:
```
def read_pkl_contents(tracename):
    """
    Reads contents of pkl file
    """
    data = []
    filename = f"{abs_root_dir}/content/CachePerformanceOnMatMul/Simulator/l
    with open(filename, 'rb') as fr:
        try:
            while True:
                data.append(pickle.load(fr))
        except EOFError:
            pass
    return data
```

In [ ]:
```
# load pkl data into object
all_data = {}
for tracename in traces:
    all_data[tracename] = read_pkl_contents(tracename)

# print(pprint.pformat(all_data))
```

In [ ]:
```
# change font size for x label
plt.rc('xtick', labelsize=4)

def convert_dict_to_list(experiments, results):
    ans = []
    for experiment in experiments:
        ans.append(results[experiment])
    return ans
```

In [ ]:
```
# plotting for each config
def plot_each_cache_data(plt_obj, data, type, title, experiments=None):
    if experiments is None :
        experiments = []
        for result in data:
            experiments.append(result["experiment_name"])

    cache_1_results = {}
    cache_2_results = {}
    cache_3_results = {}
    for result in data:
        if result["experiment_name"] in experiments:
            cache_1_results[result["experiment_name"]] = (result["caches"].g
            cache_2_results[result["experiment_name"]] = (result["caches"].g
            cache_3_results[result["experiment_name"]] = (result["caches"].g

    cache_1_results = convert_dict_to_list(experiments, cache_1_results)
    cache_2_results = convert_dict_to_list(experiments, cache_2_results)
    cache_3_results = convert_dict_to_list(experiments, cache_3_results)
    print(experiments, cache_1_results, cache_2_results, cache_3_results)

    plt_obj.plot(experiments, cache_1_results, marker='o', color='blue', lin
    plt_obj.plot(experiments, cache_2_results, marker='o', color='orange', l
    plt_obj.plot(experiments, cache_3_results, marker='o', color='green', li

    # plt_obj.bar(experiments, cache_1_results, color='blue', label=f'{type}
    # plt_obj.bar(experiments, cache_2_results, color='orange', label=f'{typ
    # plt_obj.bar(experiments, cache_3_results, color='green', label=f'{type
    plt_obj.set_title(title, loc='left')
    plt_obj.legend()

def plot_amat(plt_obj, data, type, title, experiments=None):
    if experiments is None :
        experiments = []
        for result in data:
            experiments.append(result["experiment_name"])

    cache_1_results = {}
    cache_2_results = {}
    cache_3_results = {}
    for result in data:
        if result["experiment_name"] in experiments:
            cache_1_results[result["experiment_name"]] = (result["caches"].g
            cache_2_results[result["experiment_name"]] = (result["caches"].g
            cache_3_results[result["experiment_name"]] = (result["caches"].g

    cache_1_results = convert_dict_to_list(experiments, cache_1_results)
    cache_2_results = convert_dict_to_list(experiments, cache_2_results)
    cache_3_results = convert_dict_to_list(experiments, cache_3_results)
    print(experiments, cache_1_results, cache_2_results, cache_3_results)

    # plt_obj.plot(experiments, cache_1_results, marker='o', color='blue', l
    # plt_obj.plot(experiments, cache_2_results, marker='o', color='orange',
    # plt_obj.plot(experiments, cache_3_results, marker='o', color='green',

    plt_obj.bar(experiments, cache_1_results, color='blue', label=f'{type} f
    plt_obj.bar(experiments, cache_2_results, color='orange', label=f'{type}
    plt_obj.bar(experiments, cache_3_results, color='green', label=f'{type}
```

```
    plt_obj.set_title(title, loc='left')
    plt_obj.legend()

# plotting rate for each config
def plot_rate(plt_obj, data, type, title, experiments=None):
    if experiments is None :
        experiments = []
        for result in data:
            experiments.append(result["experiment_name"])
    cache_1_results = {}
    cache_2_results = {}
    cache_3_results = {}
    for result in data:
        if result["experiment_name"] in experiments:
            cache_1_results[result["experiment_name"]] = (result["caches"].g
            cache_2_results[result["experiment_name"]] = (result["caches"].g
            cache_3_results[result["experiment_name"]] = (result["caches"].g

    cache_1_results = convert_dict_to_list(experiments, cache_1_results)
    cache_2_results = convert_dict_to_list(experiments, cache_2_results)
    cache_3_results = convert_dict_to_list(experiments, cache_3_results)
    print(experiments, cache_1_results, cache_2_results, cache_3_results)

    plt_obj.plot(experiments, cache_1_results, marker='o', color='blue', lin
    plt_obj.plot(experiments, cache_2_results, marker='o', color='orange', l
    plt_obj.plot(experiments, cache_3_results, marker='o', color='green', li
    plt_obj.set_title(title, loc='left')
    plt_obj.legend()

# plotting rate for each config
def plot_cycles(plt_obj, data, title, experiments=None):
    if experiments is None :
        experiments = []
        for result in data:
            experiments.append(result["experiment_name"])
    results = []
    for result in data:
        if result["experiment_name"] in experiments:
            results.append(result["tot_cycles"])
    print(experiments, results)
    plt_obj.plot(experiments, results, marker='o', color='blue', linestyle='
    plt_obj.set_title(title, loc='center')
```

Cache Study - Scatter Kernel

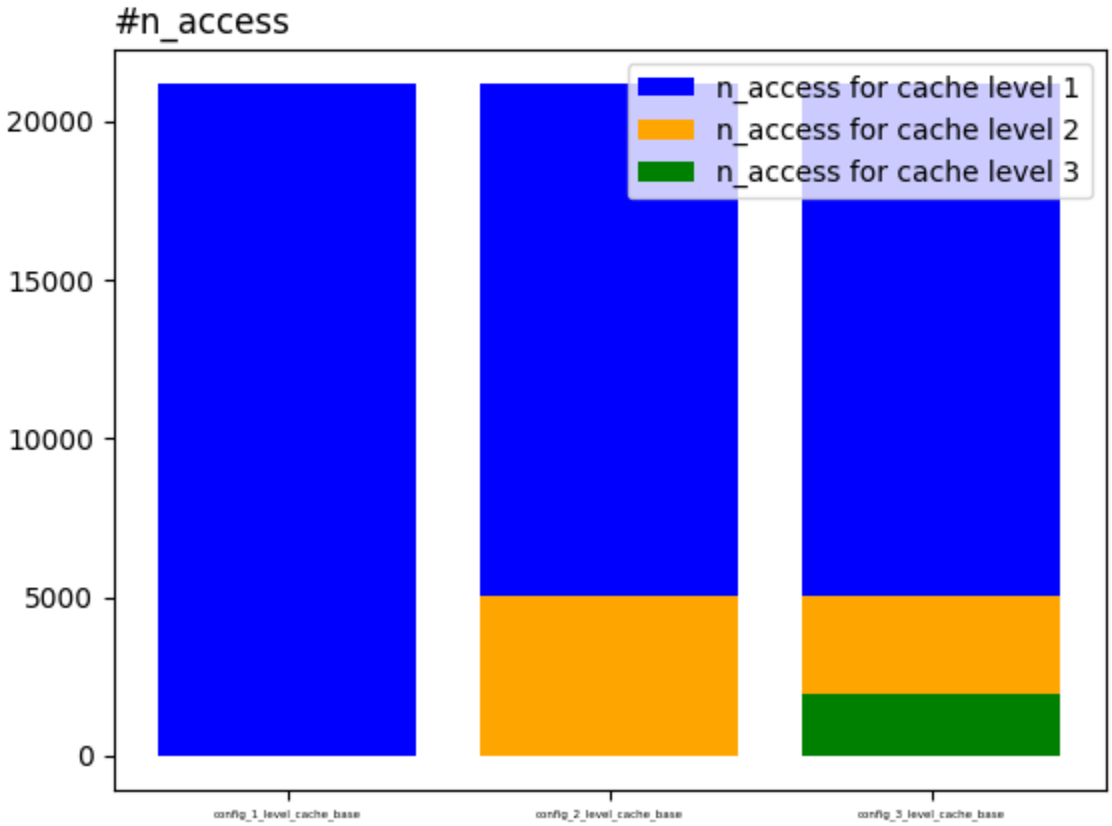
Scatter operation writes data to a given matrix from a list of indices which are random in nature. Since the operations are arbitary, i.e, no spatial and temporal locality, we expect our cache results to match with that of any general cache study.

Experiments on number of cache layers

Here we consider 3 level cache, 2 level cache and 1 level cache and plot amat vs cache layer graph

In [ ]:
```
figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_1_level_cache_base", "config_2_level_cache_base", "co
plot_amat(axis, all_data["mat_scatter"], "n_access", "#n_access", experiment
```

```
['config_1_level_cache_base', 'config_2_level_cache_base', 'config_3_level_
cache_base'] [21200, 21200, 21200] [-1, 5057, 5057] [-1, -1, 1932]
```



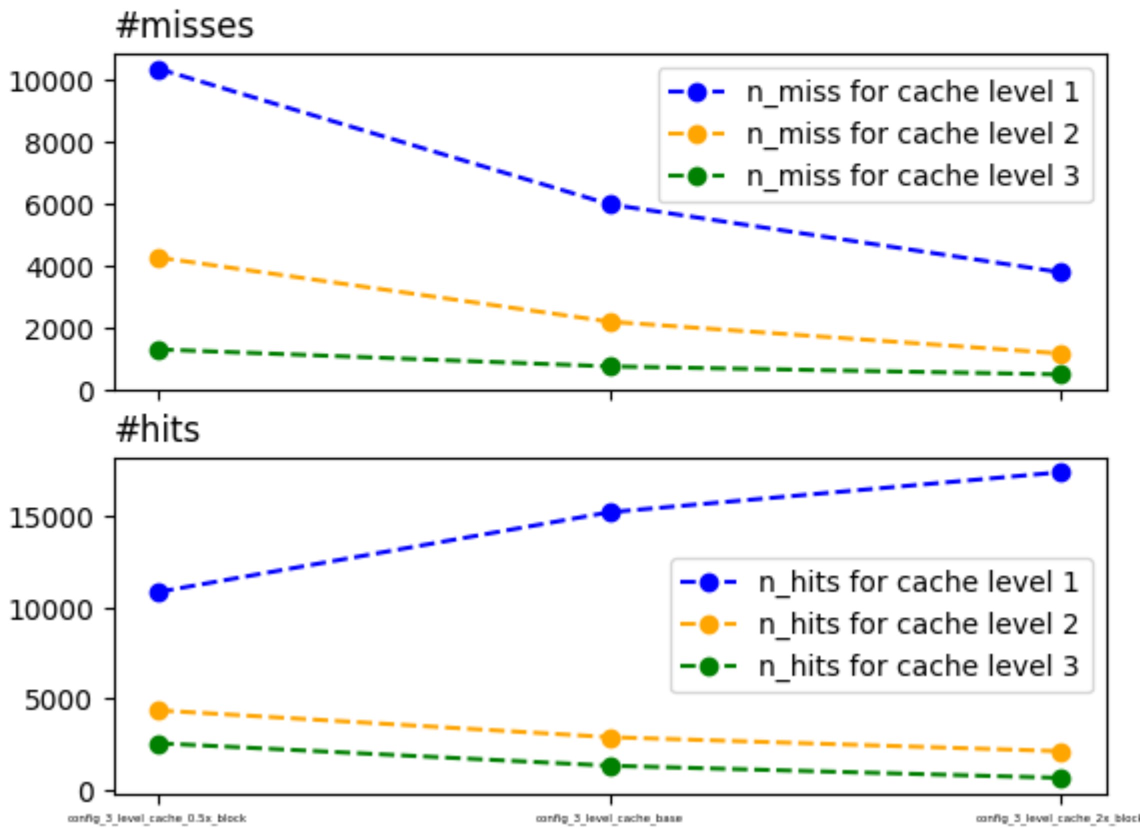
Experiments on Block size

Here we take the 3 level cache as base and reduce its block size by half, and also increase the base's block size twice.

Number of misses and hits vs Block Size

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_each_cache_data(axis[0], all_data["mat_scatter"], "n_miss", "#misses", 
plot_each_cache_data(axis[1], all_data["mat_scatter"], "n_hits", "#hits", ex
```

```
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [10359, 5982, 3785] [4263, 2190, 1162] [1293, 745, 4
85]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [10841, 15218, 17415] [4335, 2864, 2092] [2529, 129
6, 627]
```



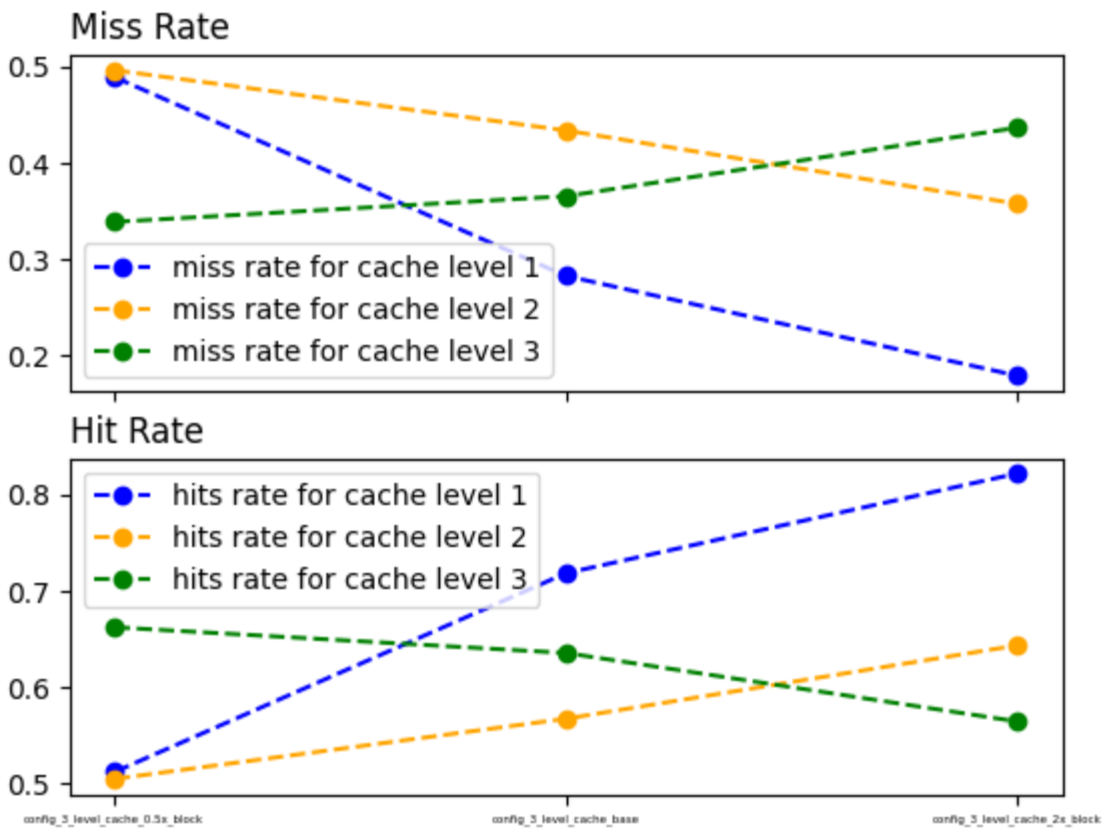
We observe the following for number of misses/hits vs block size,

- Increasing block size -> Reduces number of misses in all cache layers
- Increasing block size -> Increases number of hits in L1 cache and slightly reduces number of hits in L2 and L3 cache

Miss and hit rate vs block size

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_rate(axis[0], all_data["mat_scatter"], "n_miss", "Miss Rate", experimen
plot_rate(axis[1], all_data["mat_scatter"], "n_hits", "Hit Rate", experiment
```

```
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [0.4886320754716981, 0.2821698113207547, 0.178537735
8490566] [0.49581297976273553, 0.4333201424614167, 0.35709895513214507] [0.
33830455259026687, 0.3650171484566389, 0.43615107913669066]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [0.5113679245283019, 0.7178301886792453, 0.821462264
1509434] [0.5041870202372645, 0.5666798575385833, 0.6429010448678549] [0.66
16954474097331, 0.6349828515433611, 0.5638489208633094]
```



We observe the following for number of miss/hit rate vs block size,

- Increasing block size -> Reduces miss rate in L1 and L2 caches but increases slightly in L3 cache
- Increasing block size -> Increases hit rate in L1 and L2 cache and slightly reduces the hit rate in L3 cache

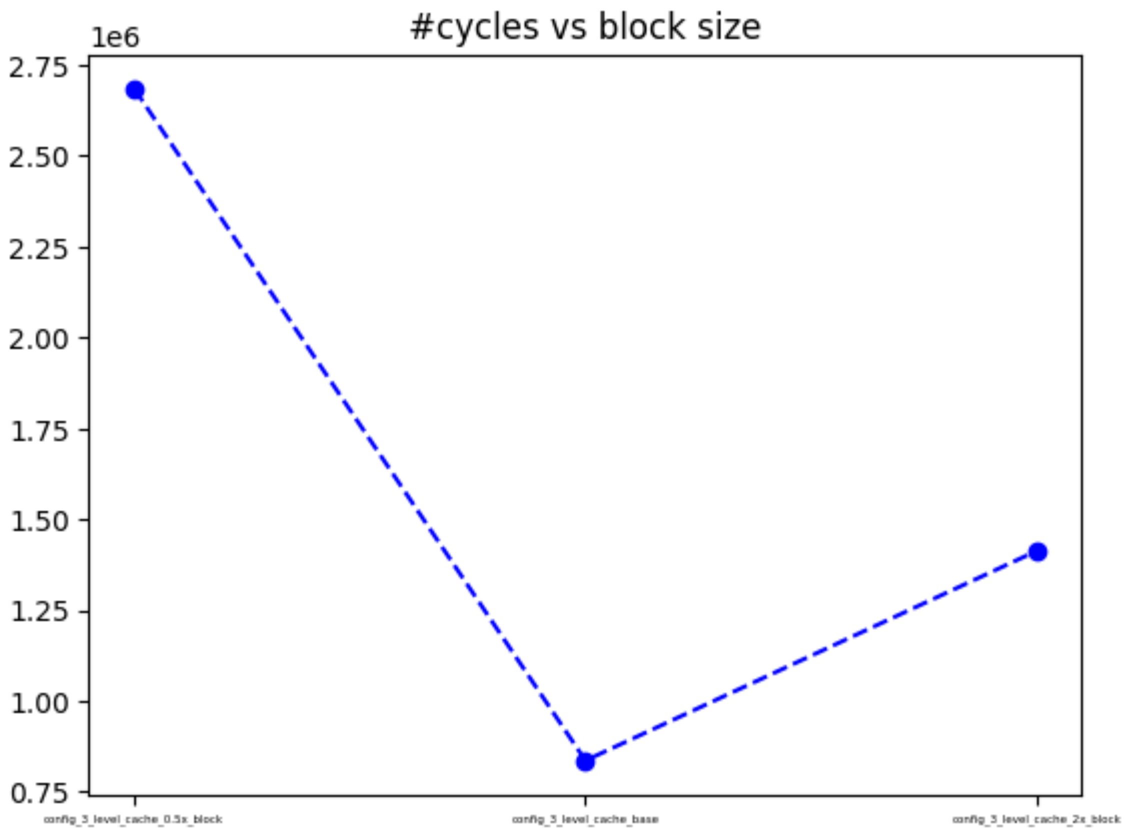
We believe the deviation in L3 layer is due to the reduced number of access to L3 cache.

Number of cycles vs block size

In [ ]:

```
figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_cycles(axis, all_data["mat_scatter"], "#cycles vs block size", experime
```

```
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [2685436, 836444, 1413368]
```



- Increasing block size -> Reduces the number of cycles taken to complete and then the #cycles increases. We conclude that the base model with block size of 16 works the best.

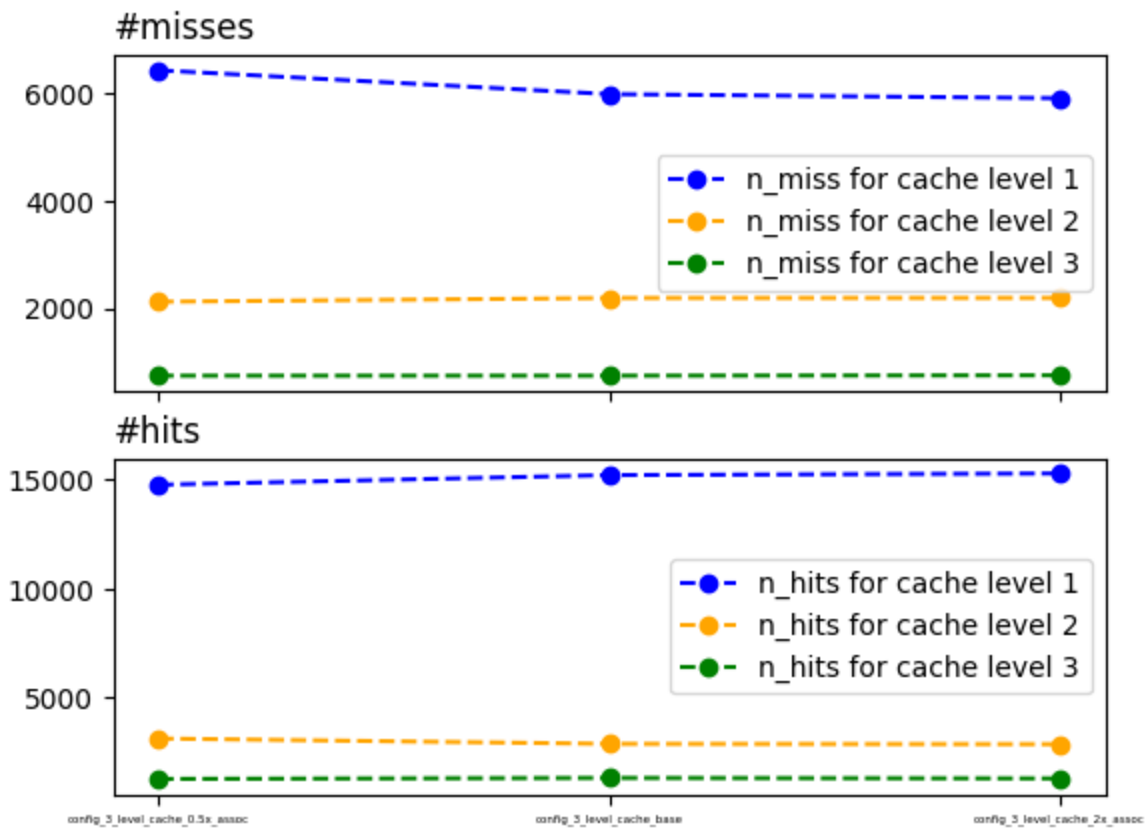
Experiments on Cache Associativity

Here we take the 3 level cache as base and reduce the associativity of L1 cache size by half, and also increase the base's L1 cache associativity twice.

Number of misses and hits vs Associativity

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_each_cache_data(axis[0], all_data["mat_scatter"], "n_miss", "#misses",
plot_each_cache_data(axis[1], all_data["mat_scatter"], "n_hits", "#hits", ex
```

```
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [6426, 5982, 5906] [2121, 2190, 2191] [745, 745, 75
2]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [14774, 15218, 15294] [3109, 2864, 2846] [1252, 129
6, 1272]
```



We observe the following for number of misses/hits vs Associativity,

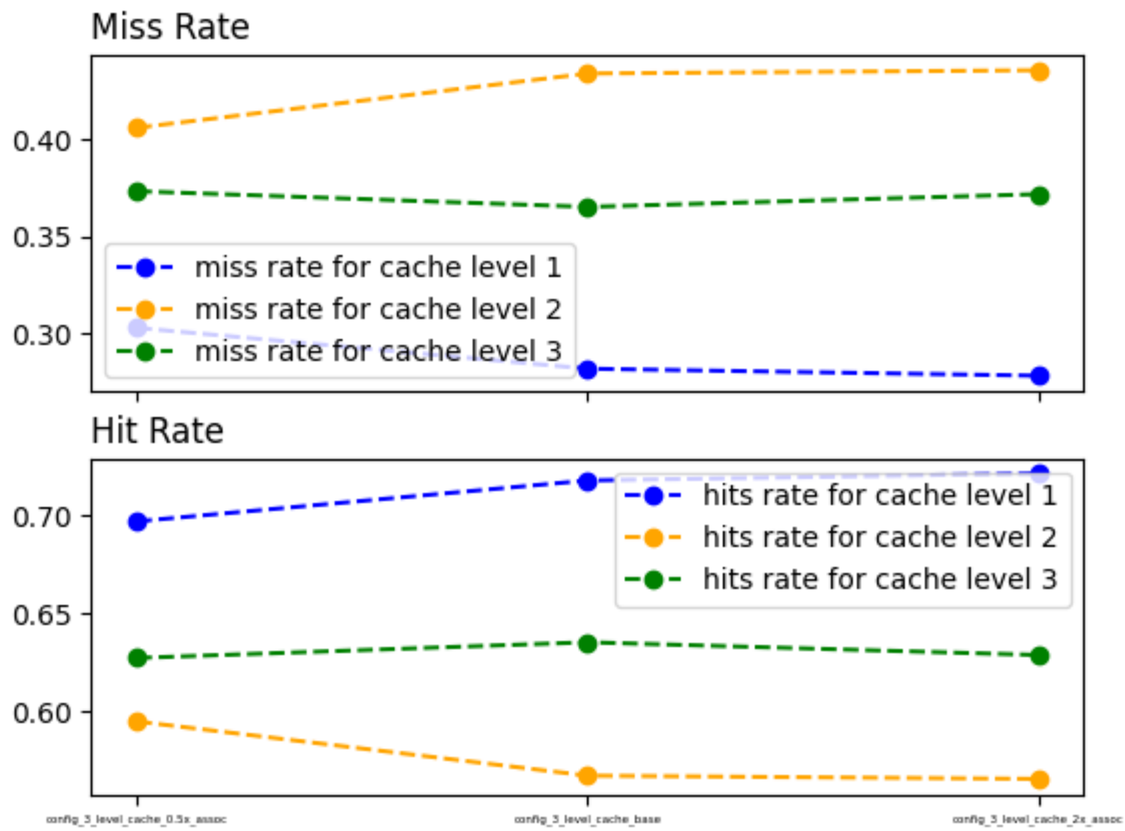
- Increasing associativity from 1 to 4 in L1 cache -> Reduces number of misses in L1 cache but this change doesn't propage to other higher layers.

The opposite is observed for number of hits.

Miss and hit rate vs Associativity

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_rate(axis[0], all_data["mat_scatter"], "n_miss", "Miss Rate", experimer
plot_rate(axis[1], all_data["mat_scatter"], "n_hits", "Hit Rate", experiment
```

```
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [0.3031132075471698, 0.2821698113207547, 0.278584905
66037736] [0.4055449330783939, 0.4333201424614167, 0.4349811395672027] [0.3
7305958938407613, 0.3650171484566389, 0.3715415019762846]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [0.6968867924528301, 0.7178301886792453, 0.721415094
3396226] [0.5944550669216061, 0.5666798575385833, 0.5650188604327973] [0.62
69404106159239, 0.6349828515433611, 0.6284584980237155]
```

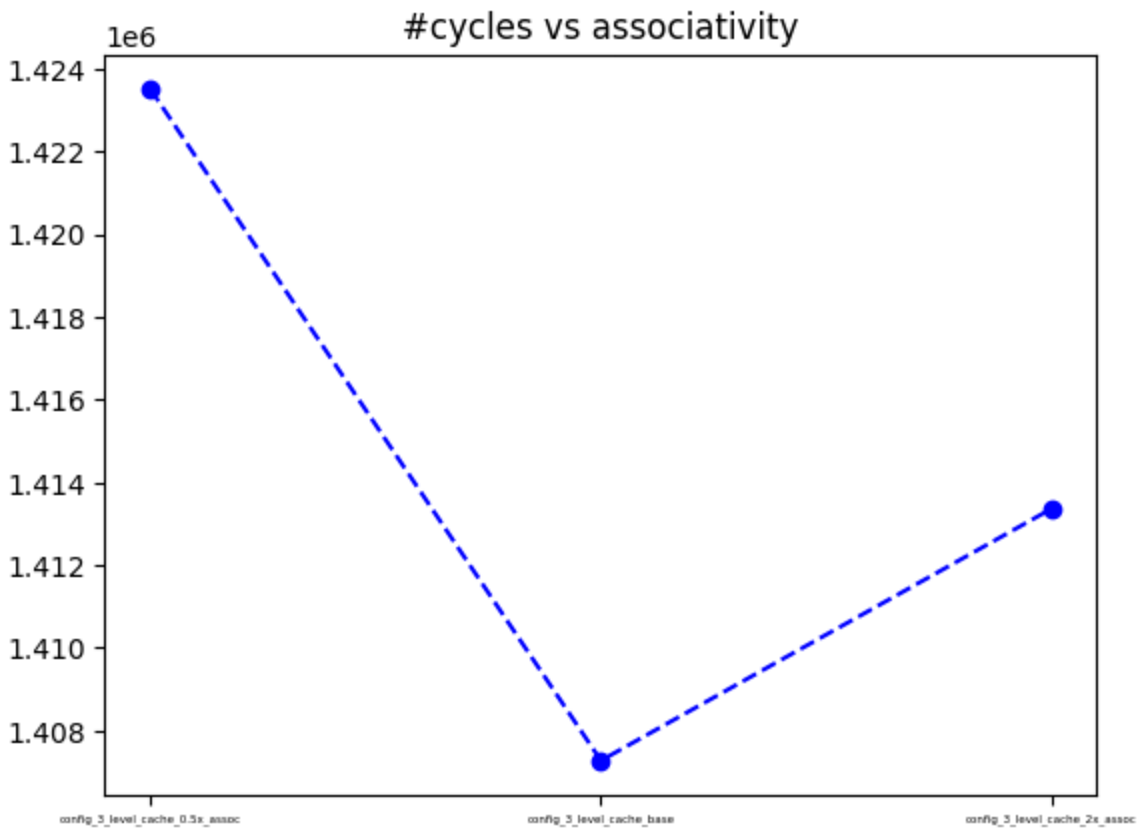


Increasing associativity of L1 cache, reduces miss rate in L1. However, a similar observation as that of above is found here, i.e, miss rate is compensated in higher layers (L2, L3)

Number of cycles vs Associativity

In [ ]:
```
figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_cycles(axis, all_data["mat_scatter"], "#cycles vs associativity", exper
```

```
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [1423520, 1407276, 1413368]
```



- Increasing associativity size -> Reduces the number of cycles taken to complete and then the #cycles increases. We conclude that the base model with associativity of 2 runs in least number of cycles.

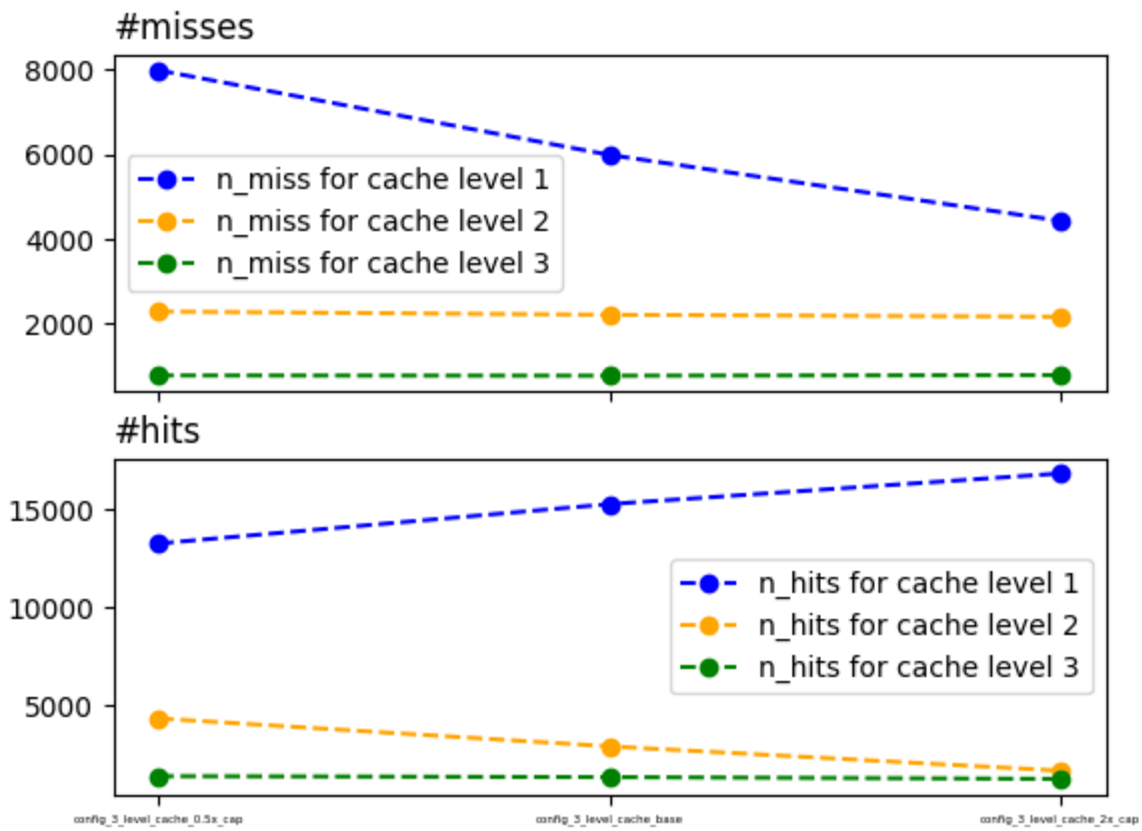
Experiments on Cache Capacity

Here we take the 3 level cache as base and reduce the L1 cache capacity by half, and also increase the base's L1 cache capacity twice.

Number of misses and hits vs Cache Capacity

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_each_cache_data(axis[0], all_data["mat_scatter"], "n_miss", "#misses",
plot_each_cache_data(axis[1], all_data["mat_scatter"], "n_hits", "#hits", ex
```

```
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [7990, 5982, 4419] [2267, 2190, 2145] [751, 745, 758]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [13210, 15218, 16781] [4288, 2864, 1623] [1345, 1296, 12
04]
```



We observe the following for number of misses/hits vs Cache Capacity,

- Increasing Cache Capacity from 8 to 32 in L1 cache -> Reduces number of misses in L1 cache by a huge margin but this change doesn't propage to other higher layers.

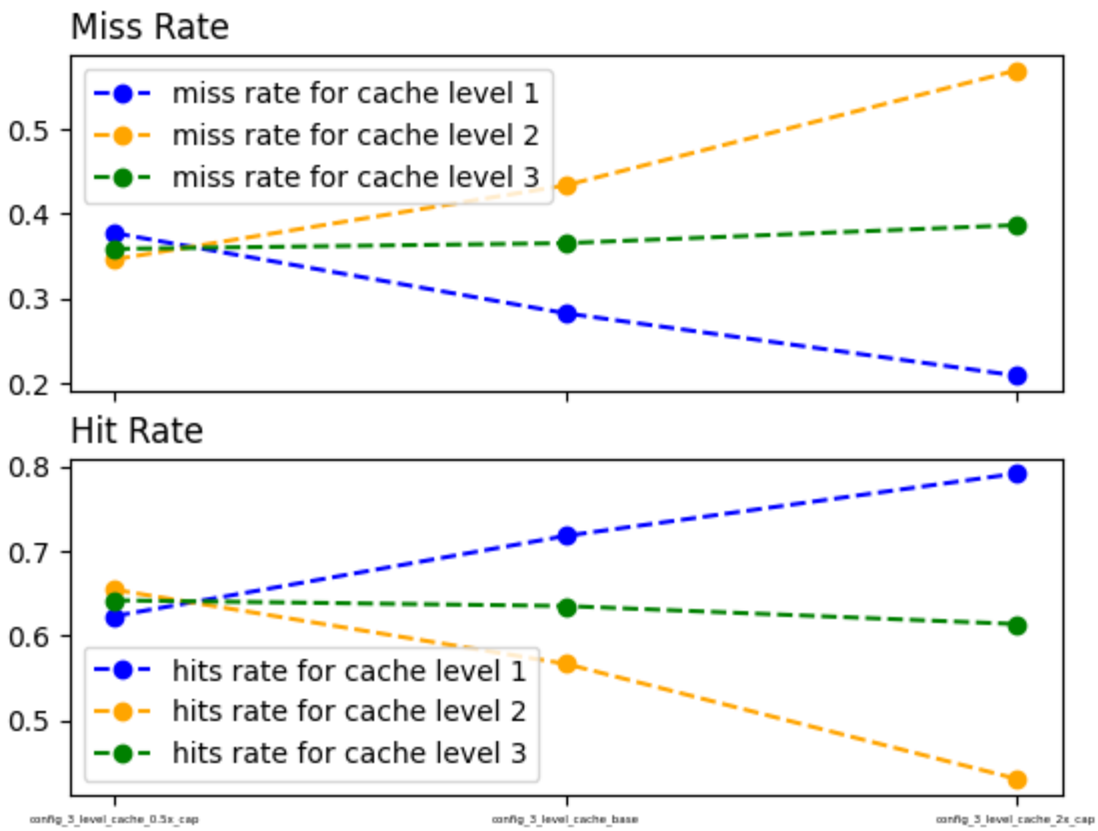
The opposite is observed for number of hits.

Miss and hit rate vs Cache Capacity

In [ ]:

```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_rate(axis[0], all_data["mat_scatter"], "n_miss", "Miss Rate", experimer
plot_rate(axis[1], all_data["mat_scatter"], "n_hits", "Hit Rate", experiment
```

```
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [0.3768867924528302, 0.2821698113207547, 0.2084433962264
1508] [0.34584286803966435, 0.4333201424614167, 0.5692675159235668] [0.3583
015267175573, 0.3650171484566389, 0.38634046890927626]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [0.6231132075471698, 0.7178301886792453, 0.7915566037735
849] [0.6541571319603356, 0.5666798575385833, 0.4307324840764331] [0.641698
4732824428, 0.6349828515433611, 0.6136595310907238]
```



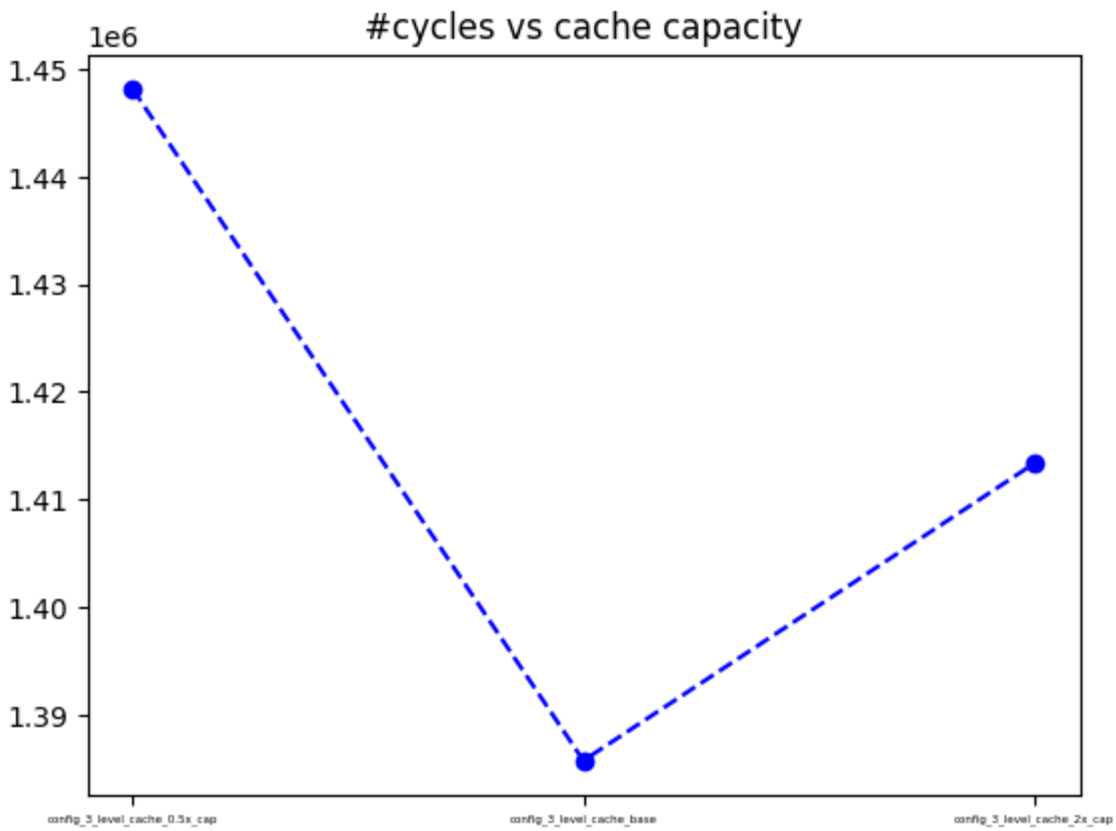
- Increasing L1 cache capacity improves the hit rate. However, L2 and L3 suffer a loss in hit rate. We believe that this is due to cache capacity of L1 being almost similar to that of L2.

Thus, we can conclude that the L2, L3 layer should have higher cache capacity when compared to L1 in order to optimize for hit rate

Number of cycles vs Cache Capacity

In [ ]:
```
figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_cycles(axis, all_data["mat_scatter"], "#cycles vs cache capacity", expe
```

```
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [1448168, 1385748, 1413368]
```



Cache Study - Gather Kernel

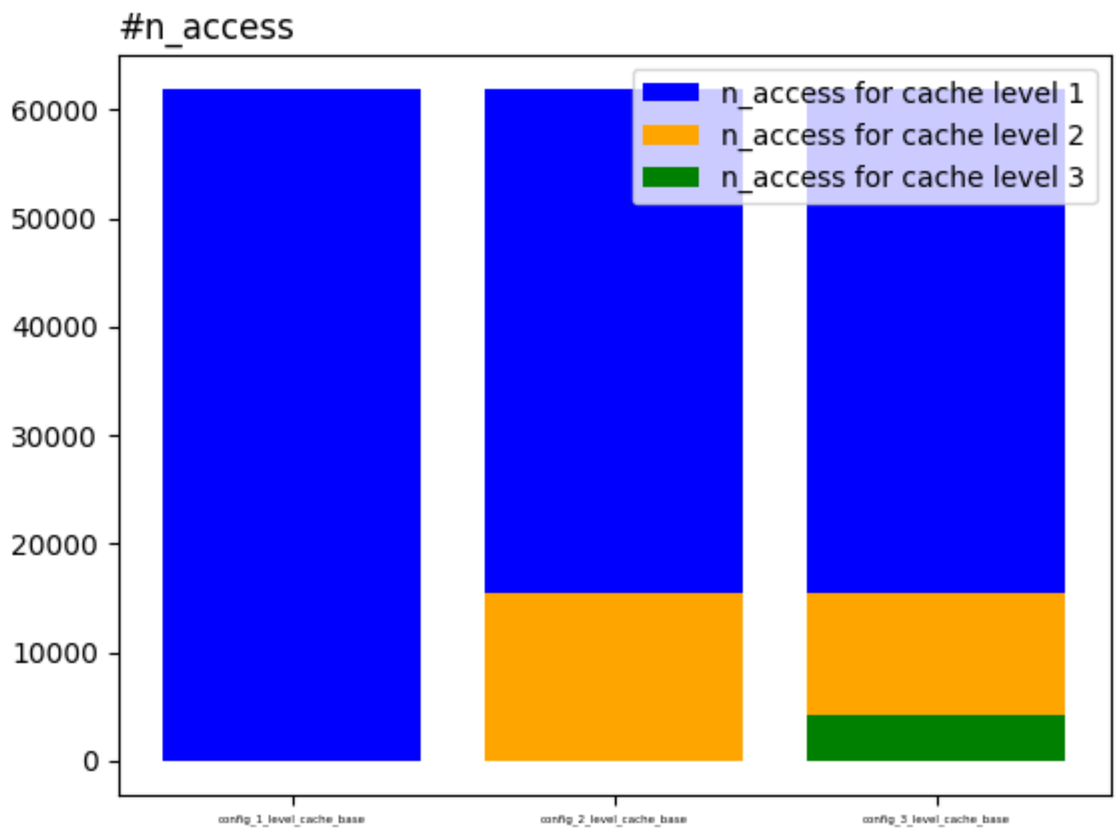
Gather operation reads data from random indices of a given matrix. Since the operations are arbitary, i.e, no spatial and temporal locality, we expect our cache results to match with that of general cache study.

Experiments on number of cache layers

Here we consider 3 level cache, 2 level cache and 1 level cache and plot amat vs cache layer graph

In [ ]:
```
figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_1_level_cache_base", "config_2_level_cache_base", "co
plot_amat(axis, all_data["mat_gather"], "n_access", "#n_access", experiments
```

```
['config_1_level_cache_base', 'config_2_level_cache_base', 'config_3_level_
cache_base'] [61957, 61957, 61957] [-1, 15406, 15406] [-1, -1, 4179]
```

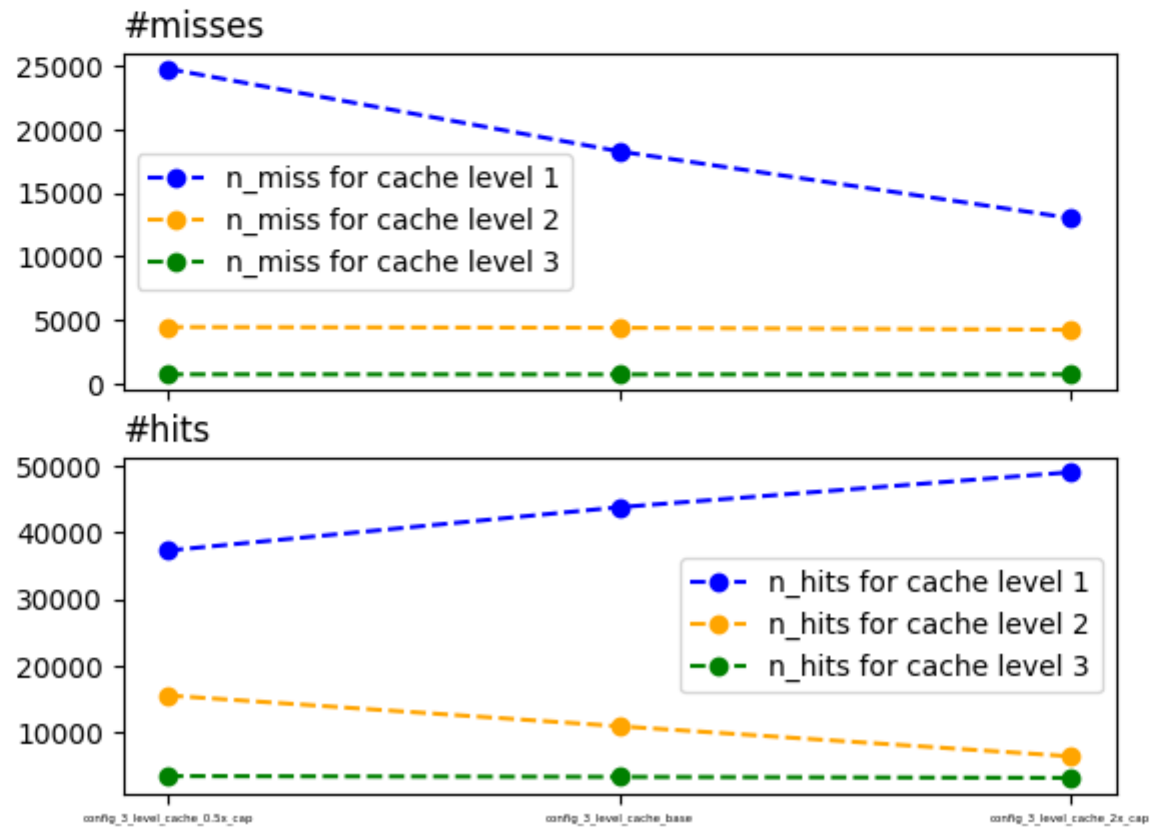


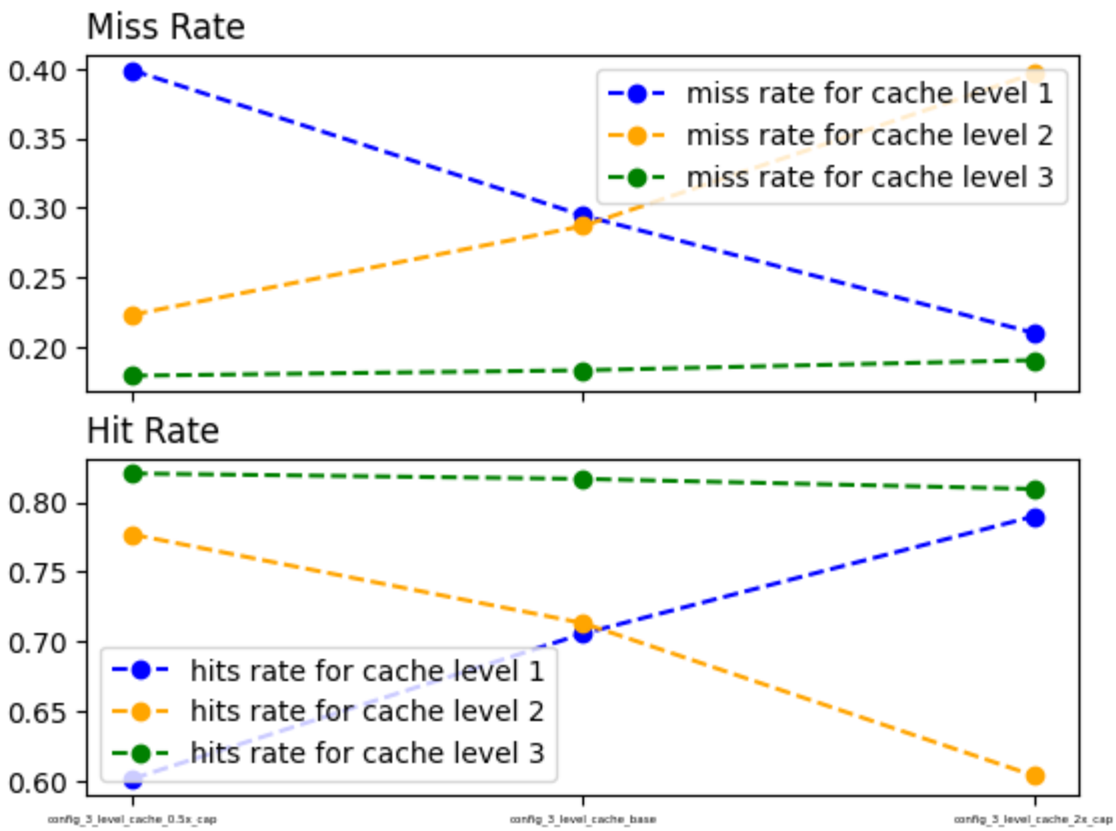
Experiments on Cache Capacity

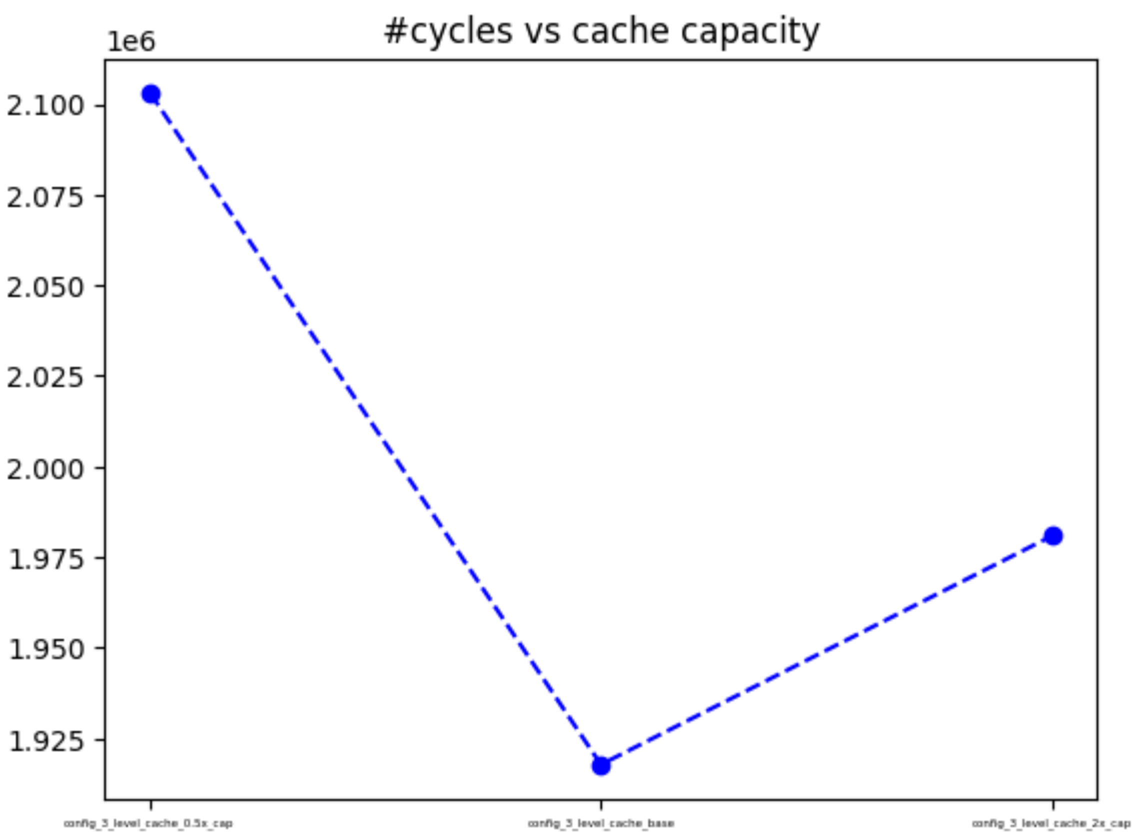
Here we take the 3 level cache as base and reduce the L1 cache capacity by half, and also increase the base's L1 cache capacity twice.

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_each_cache_data(axis[0], all_data["mat_gather"], "n_miss", "#misses", e
plot_each_cache_data(axis[1], all_data["mat_gather"], "n_hits", "#hits", exp

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_rate(axis[0], all_data["mat_gather"], "n_miss", "Miss Rate", experiment
plot_rate(axis[1], all_data["mat_gather"], "n_hits", "Hit Rate", experiments

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_cycles(axis, all_data["mat_gather"], "#cycles vs cache capacity", exper
```

```
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [24711, 18243, 13016] [4477, 4421, 4265] [773, 766, 769]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [37246, 43714, 48941] [15583, 10985, 6489] [3536, 3413,
3267]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [0.39884113175266717, 0.29444614813499687, 0.21008118533
821843] [0.2231804586241276, 0.28696611709723485, 0.39659661521294404] [0.1
7939197029473195, 0.18329743957884662, 0.19053518334985134]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [0.6011588682473328, 0.7055538518650032, 0.7899188146617
816] [0.7768195413758724, 0.7130338829027651, 0.603403384787056] [0.8206080
29705268, 0.8167025604211534, 0.8094648166501487]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [2103197, 1917753, 1980873]
```




Miss Rate
0.40
0.35
0.30
0.25
0.20
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
Hit Rate
0.80
0.75
0.70
0.65
0.60
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_cap
config_3_level_cache_base
config_3_level_cache_2x_cap

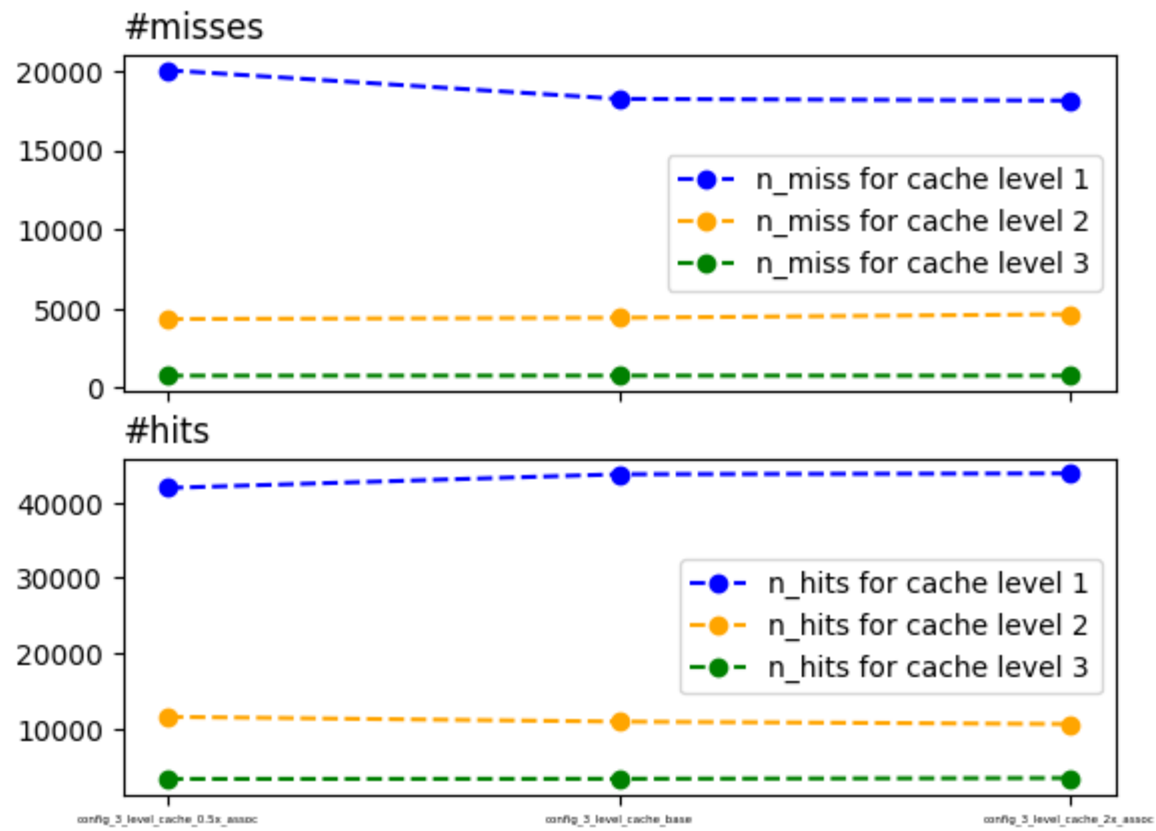

#cycles vs cache capacity
1e6
2.100
2.075
2.050
2.025
2.000
1.975
1.950
1.925
config_3_level_cache_0.5x_cap
config_3_level_cache_base
config_3_level_cache_2x_cap

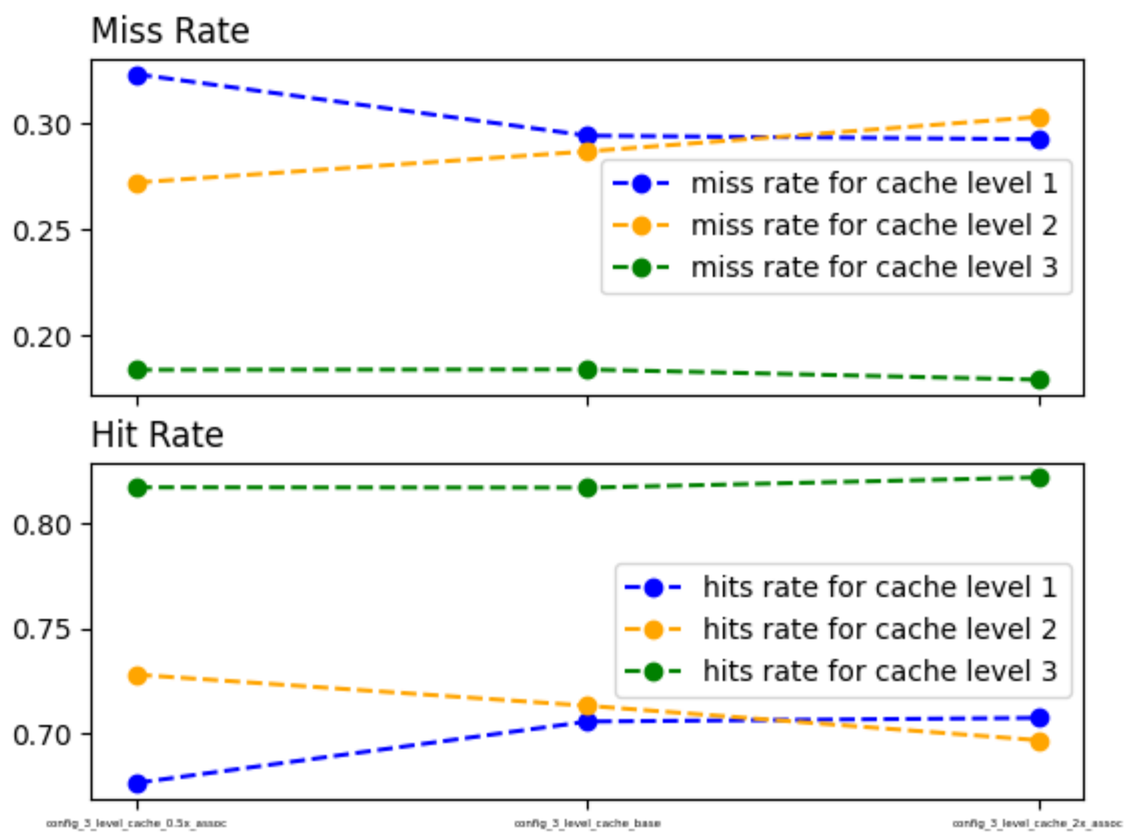
Experiments on Cache Associativity

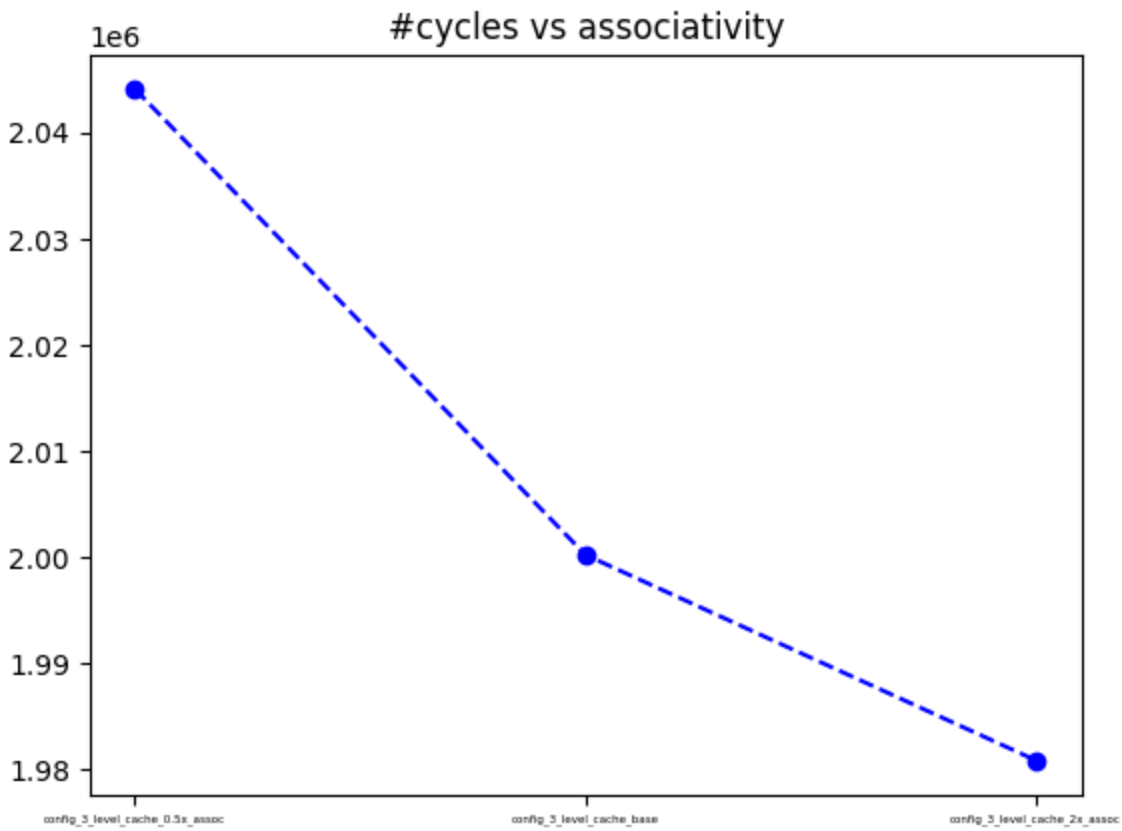
Here we take the 3 level cache as base and reduce the associativity of L1 cache size by half, and also increase the base's L1 cache associativity twice.

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_each_cache_data(axis[0], all_data["mat_gather"], "n_miss", "#misses", e
plot_each_cache_data(axis[1], all_data["mat_gather"], "n_hits", "#hits", exp

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_rate(axis[0], all_data["mat_gather"], "n_miss", "Miss Rate", experiment
plot_rate(axis[1], all_data["mat_gather"], "n_hits", "Hit Rate", experiments

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_cycles(axis, all_data["mat_gather"], "#cycles vs associativity", experi
```

```
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [20051, 18243, 18136] [4342, 4421, 4642] [762, 766,
763]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [41906, 43714, 43821] [11611, 10985, 10660] [3399, 3
413, 3513]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [0.32362767726003516, 0.29444614813499687, 0.2927191
439223978] [0.2721745126308531, 0.28696611709723485, 0.3033590380342439]
[0.18312905551550107, 0.18329743957884662, 0.1784377923292797]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [0.6763723227399648, 0.7055538518650032, 0.707280856
0776022] [0.7278254873691469, 0.7130338829027651, 0.6966409619657561] [0.81
68709444844989, 0.8167025604211534, 0.8215622076707203]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [2044105, 2000229, 1980873]
```


#misses
20000
15000
10000
5000
0
n_miss for cache level 1
n_miss for cache level 2
n_miss for cache level 3
#hits
40000
30000
20000
10000
n_hits for cache level 1
n_hits for cache level 2
n_hits for cache level 3
config_3_level_cache_0.5x_assoc
config_3_level_cache_base
config_3_level_cache_2x_assoc


Miss Rate
0.30
0.25
0.20
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
Hit Rate
0.80
0.75
0.70
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_assoc
config_3_level_cache_base
config_3_level_cache_2x_assoc

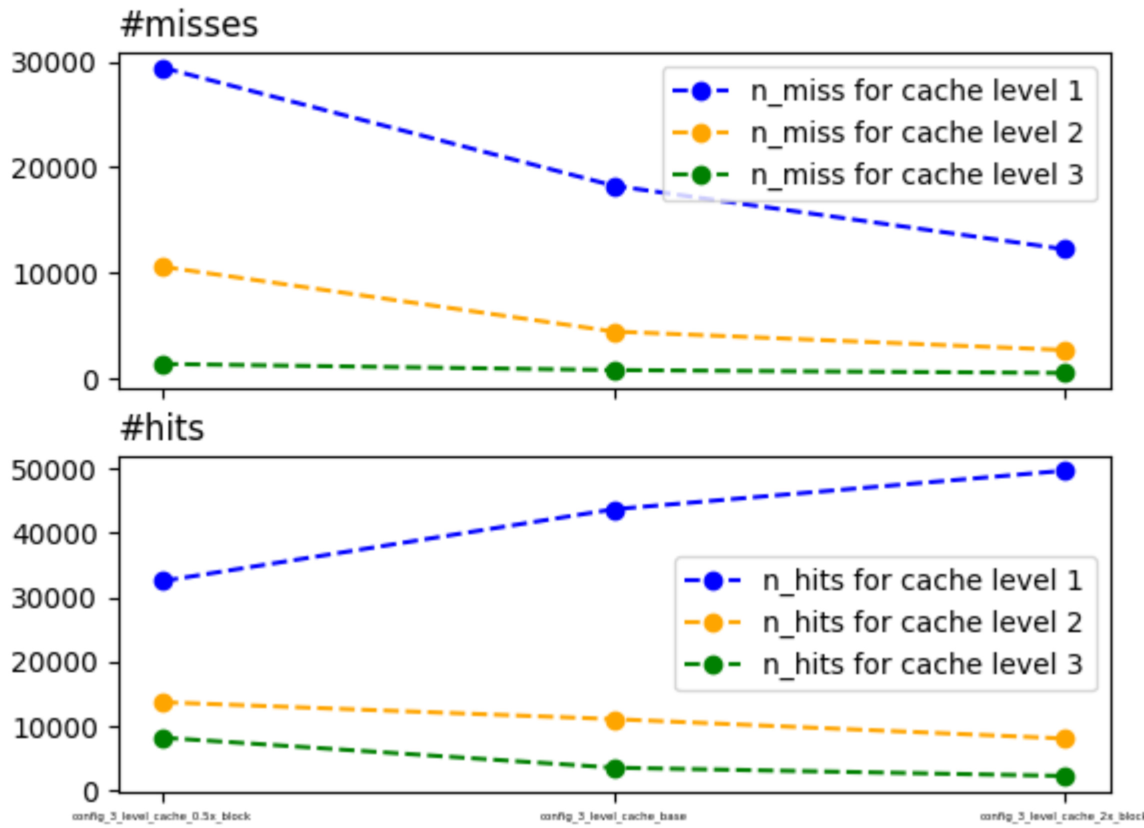


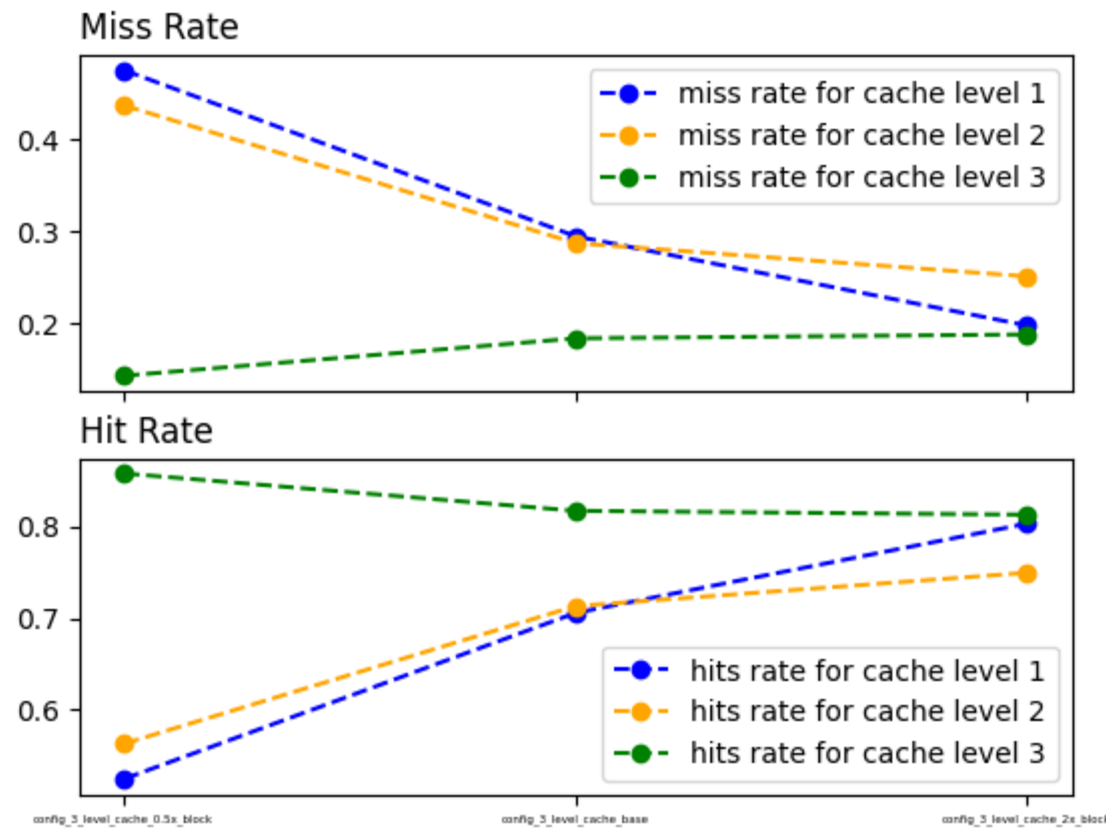
Experiments on Block size

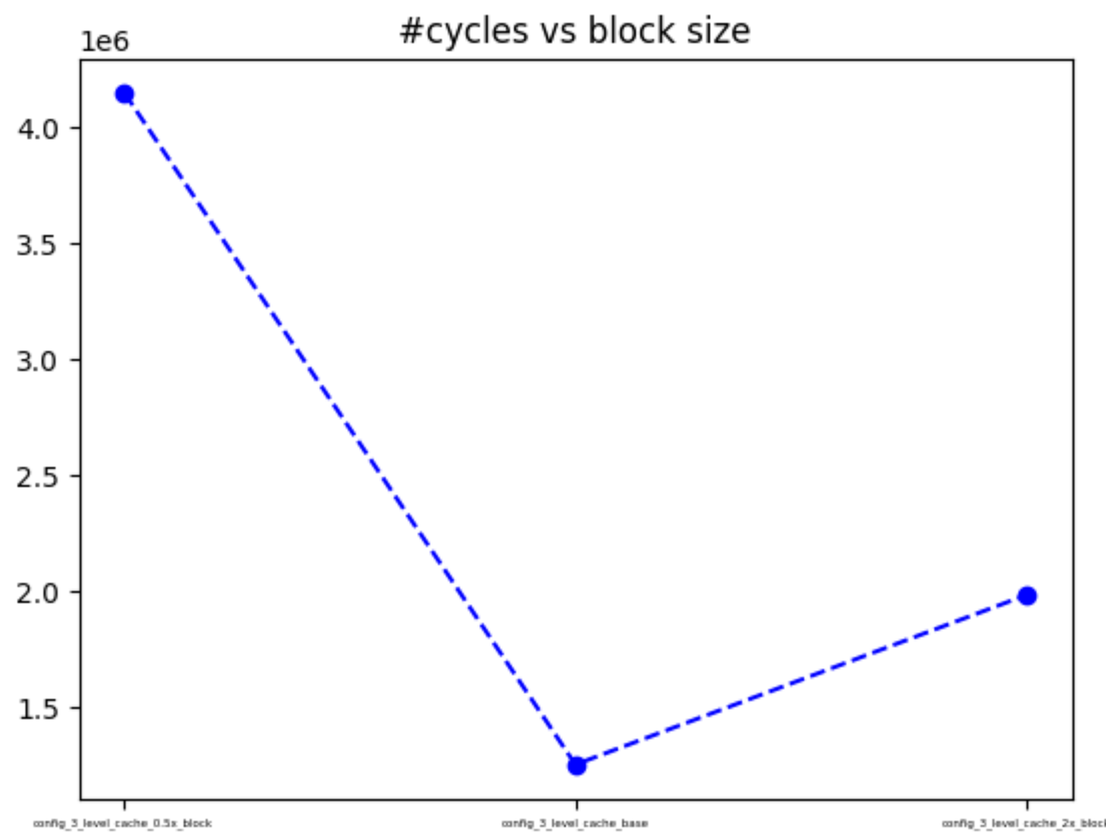
Here we take the 3 level cache as base and reduce its block size by half, and also increase the base's block size twice.

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_each_cache_data(axis[0], all_data["mat_gather"], "n_miss", "#misses", e
plot_each_cache_data(axis[1], all_data["mat_gather"], "n_hits", "#hits", exp

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_rate(axis[0], all_data["mat_gather"], "n_miss", "Miss Rate", experiment
plot_rate(axis[1], all_data["mat_gather"], "n_hits", "Hit Rate", experiments

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_cycles(axis, all_data["mat_gather"], "#cycles vs block size", experimer
```

```
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [29431, 18243, 12225] [10561, 4421, 2664] [1346, 76
6, 490]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [32526, 43714, 49732] [13623, 10985, 7961] [8090, 34
13, 2123]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [0.4750229998224575, 0.29444614813499687, 0.19731426
634601418] [0.4366936817730731, 0.28696611709723485, 0.25072941176470587]
[0.14264518863925393, 0.18329743957884662, 0.18752391886720246]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [0.5249770001775425, 0.7055538518650032, 0.802685733
6539859] [0.5633063182269269, 0.7130338829027651, 0.7492705882352941] [0.85
73548113607461, 0.8167025604211534, 0.8124760811327976]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [4146517, 1249213, 1980873]
```




Miss Rate
0.4
0.3
0.2
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
Hit Rate
0.8
0.7
0.6
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_block
config_3_level_cache_base
config_3_level_cache_2x_block


#cycles vs block size
1e6
4.0
3.5
3.0
2.5
2.0
1.5
config_3_level_cache_0.5x_block
config_3_level_cache_base
config_3_level_cache_2x_block

We observe the following for gather kernel after playing around with the cache parameters

- Increasing block size from 8 to 32.
  - Increases #hits, hit rate for L1 cache
  - Decreases #cycles and then increases
- Increasing associativity from 1 to 4
  - Reduces #misses, miss rate for L1 cache
  - Decreases #cycles and then increases
- Increasing cache capacity from 8 to 32
  - Increases #hits, hit rates
  - Decreases #cycles and then increases

Cache Study - Transpose Kernel

Transpose operation reads exchanges columns and rows of a given matrix. We implement a naive kernel in this case and observe the effects of temporal and spatial locality. We theorize that the naive implementation, having minimal spatial and temporal locality will not be cache friendly.

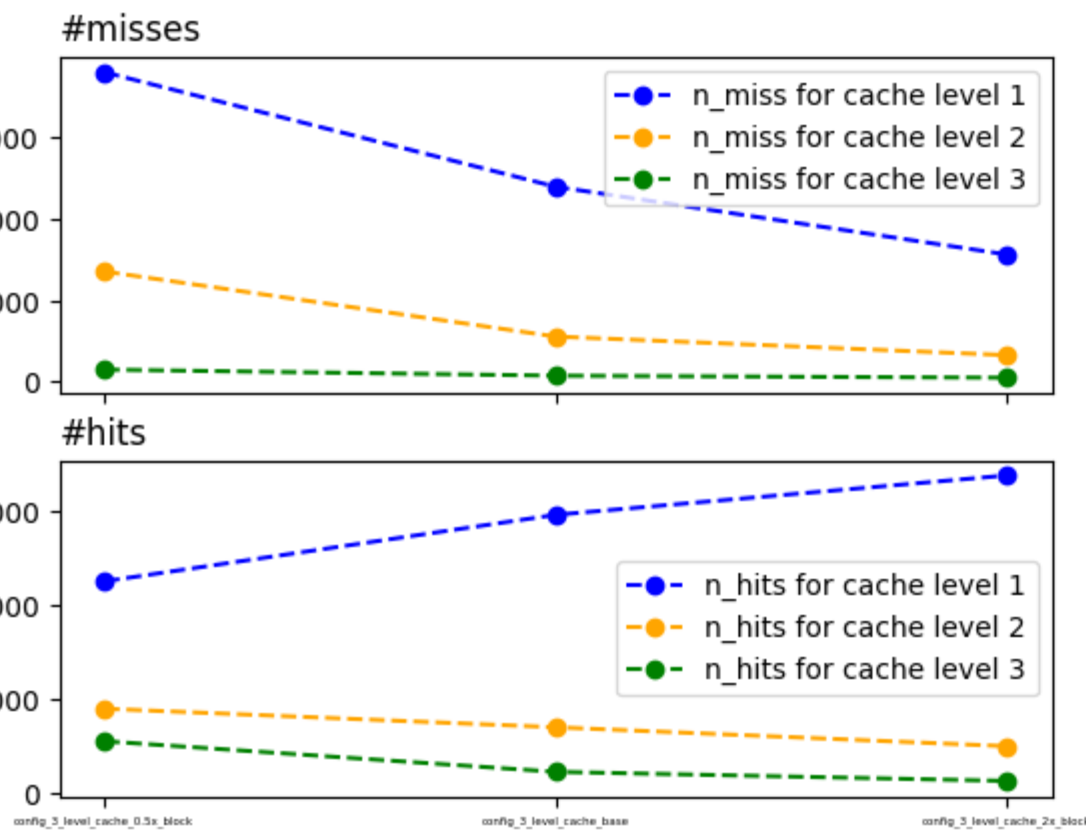
Experiments on Block size

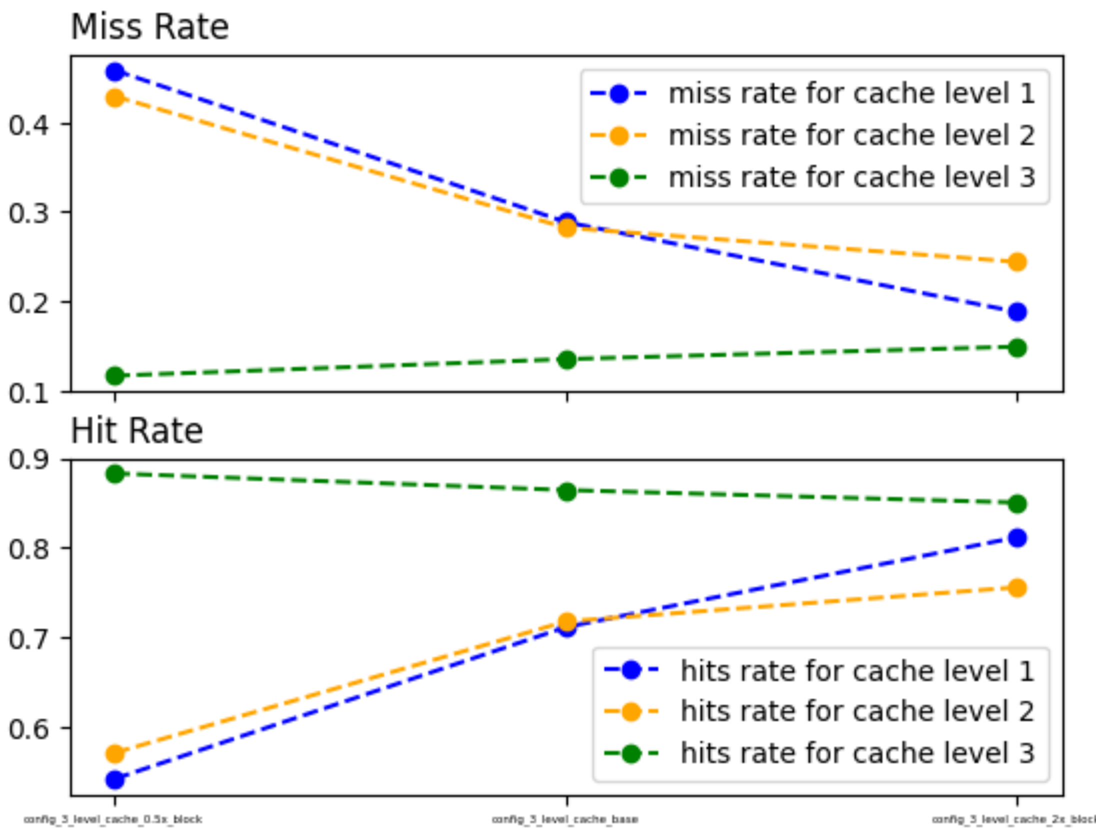
Here we take the 3 level cache as base and reduce its block size by half, and also increase the base's block size twice.

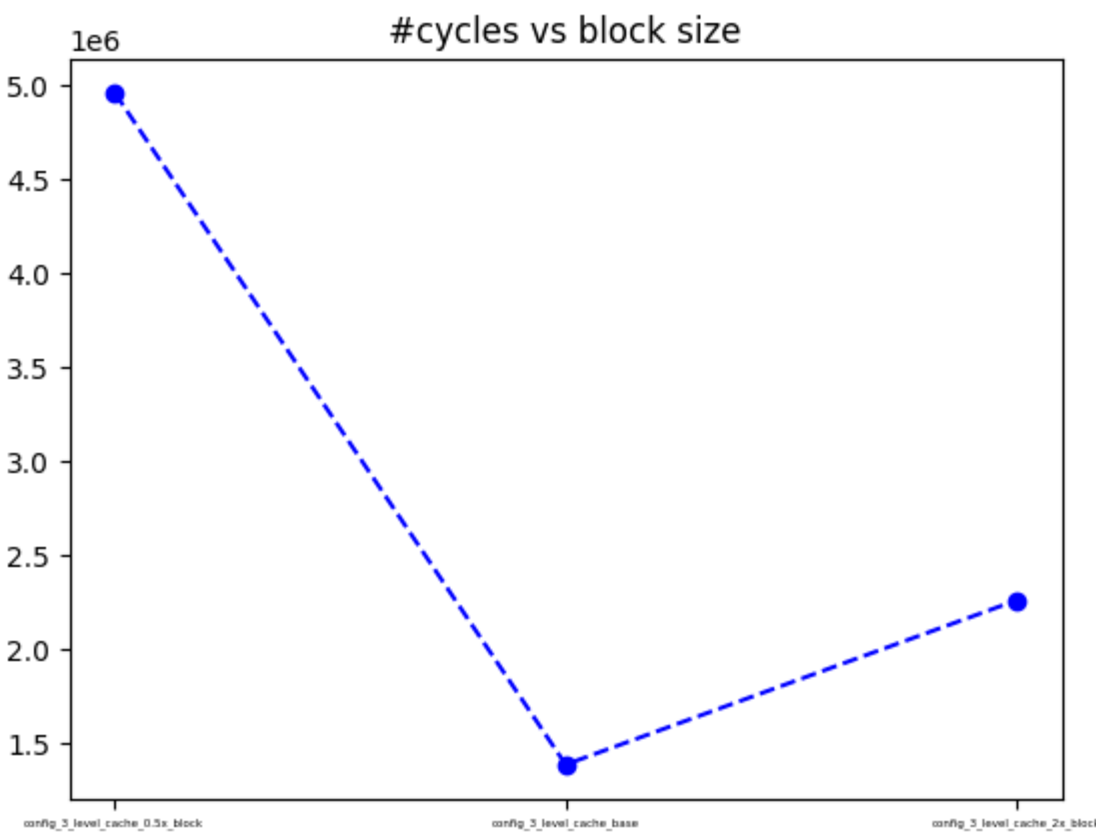
In [ ]:

```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_each_cache_data(axis[0], all_data["mat_transpose"], "n_miss", "#misses"
plot_each_cache_data(axis[1], all_data["mat_transpose"], "n_hits", "#hits",

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_rate(axis[0], all_data["mat_transpose"], "n_miss", "Miss Rate", experim
plot_rate(axis[1], all_data["mat_transpose"], "n_hits", "Hit Rate", experime

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_cycles(axis, all_data["mat_transpose"], "#cycles vs block size", experi
```

```
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [38073, 23990, 15646] [13551, 5545, 3258] [1476, 72
9, 479]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [44994, 59077, 67421] [18017, 14088, 10079] [11134,
4648, 2725]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [0.4583408573801871, 0.2888030144341315, 0.188353979
31790002] [0.4292638114546376, 0.2824326389242602, 0.24428282222388842] [0.
11704996034892942, 0.13557745954993491, 0.14950062421972535]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [0.541659142619813, 0.7111969855658685, 0.8116460206
821] [0.5707361885453623, 0.7175673610757398, 0.7557171777761116] [0.882950
0396510706, 0.8644225404500651, 0.8504993757802747]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [4961291, 1389719, 2263359]
```




Miss Rate
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
0.4
0.3
0.2
0.1
Hit Rate
0.9
0.8
0.7
0.6
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_block
config_3_level_cache_base
config_3_level_cache_2x_block

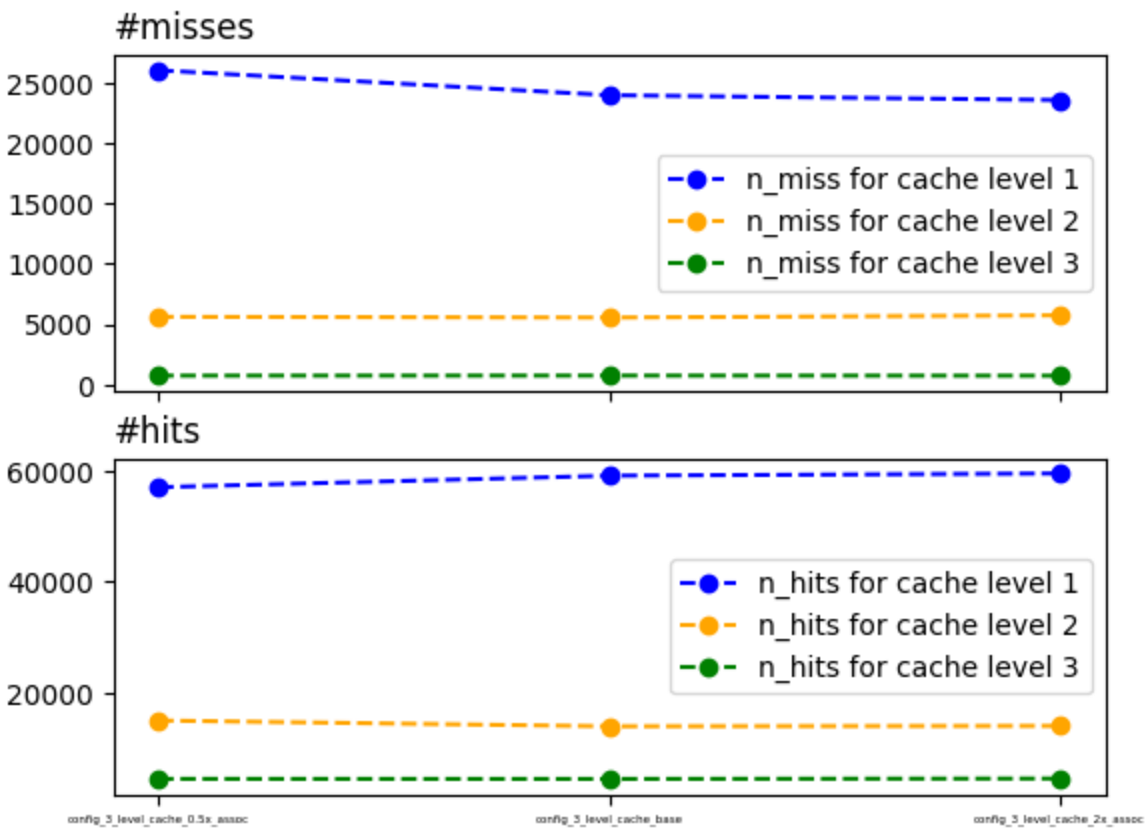

#cycles vs block size
1e6
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
config_3_level_cache_0.5x_block
config_3_level_cache_base
config_3_level_cache_2x_block

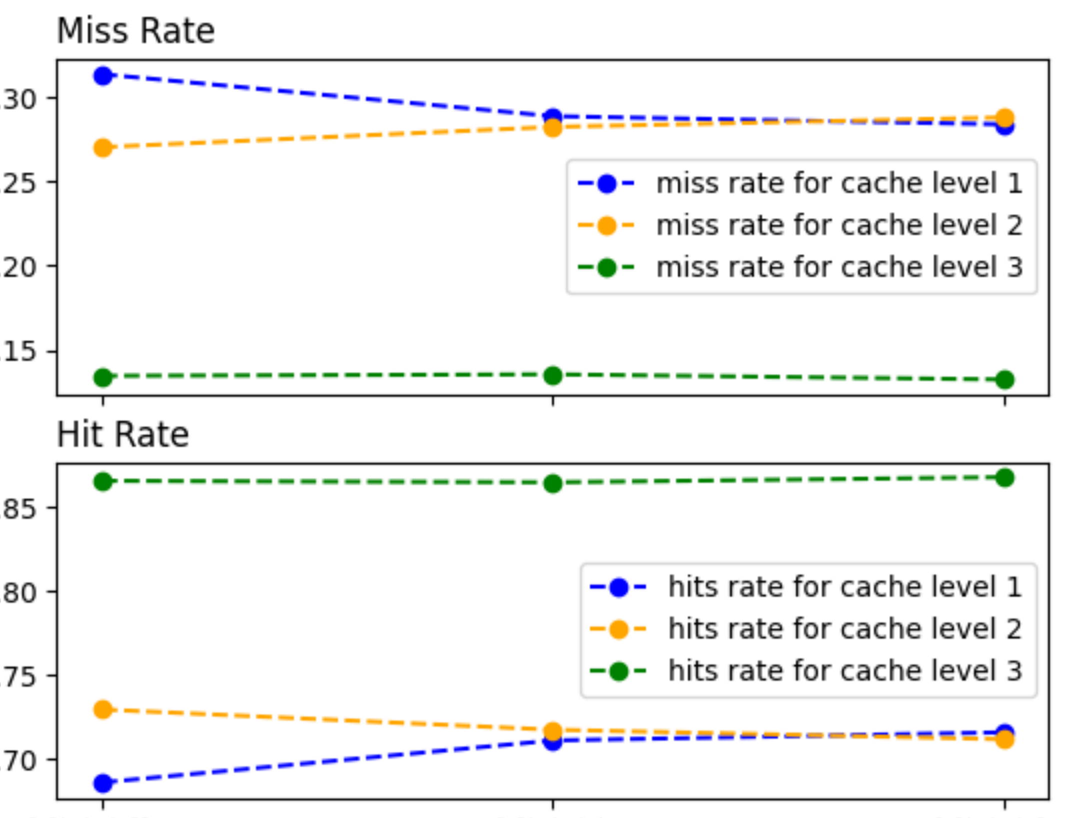
Experiments on Cache Associativity

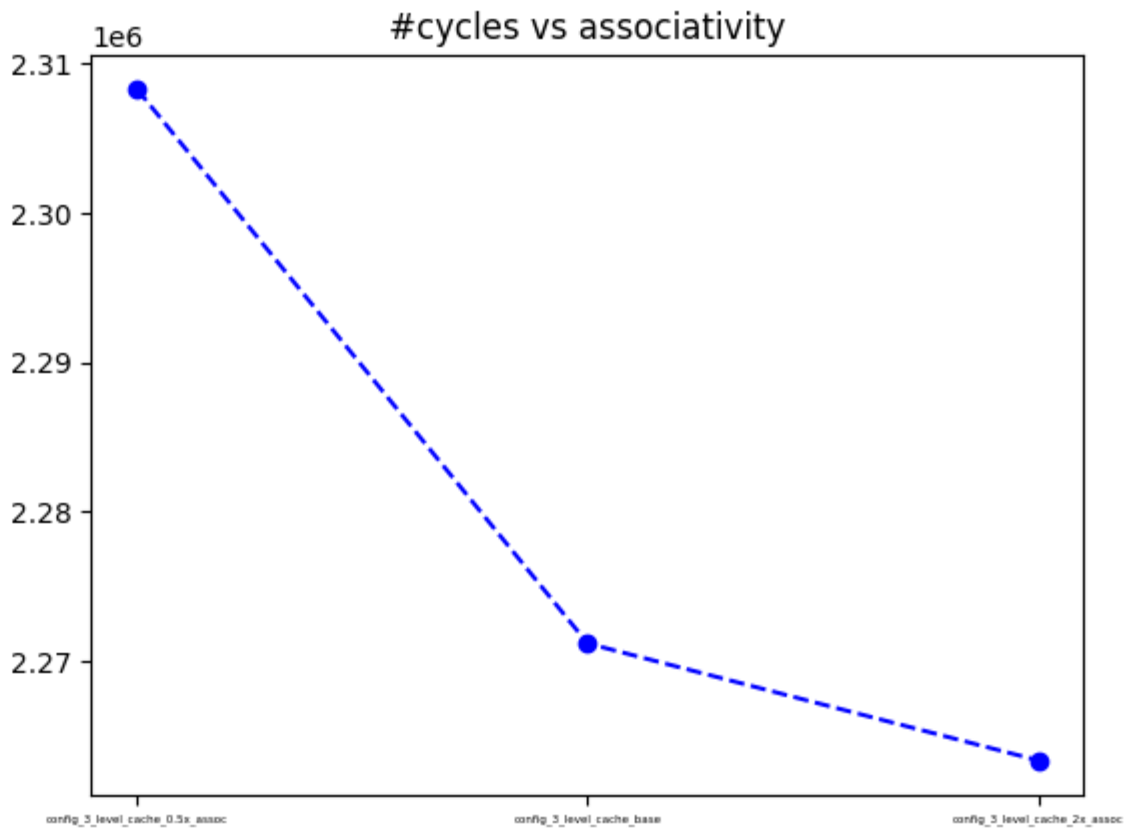
Here we take the 3 level cache as base and reduce the associativity of L1 cache size by half, and also increase the base's L1 cache associativity twice.

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_each_cache_data(axis[0], all_data["mat_transpose"], "n_miss", "#misses"
plot_each_cache_data(axis[1], all_data["mat_transpose"], "n_hits", "#hits",

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_rate(axis[0], all_data["mat_transpose"], "n_miss", "Miss Rate", experim
plot_rate(axis[1], all_data["mat_transpose"], "n_hits", "Hit Rate", experime

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_cycles(axis, all_data["mat_transpose"], "#cycles vs associativity", exp
```

```
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [26060, 23990, 23590] [5608, 5545, 5730] [725, 729,
720]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [57007, 59077, 59477] [15130, 14088, 14161] [4658, 4
648, 4713]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [0.31372265761373347, 0.2888030144341315, 0.28398762
444773495] [0.2704214485485582, 0.2824326389242602, 0.2880699813986225] [0.
1346832621214936, 0.13557745954993491, 0.13252346769740475]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [0.6862773423862665, 0.7111969855658685, 0.716012375
552265] [0.7295785514514418, 0.7175673610757398, 0.7119300186013775] [0.865
3167378785064, 0.8644225404500651, 0.8674765323025952]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [2308291, 2271191, 2263359]
```


#misses
n_miss for cache level 1
n_miss for cache level 2
n_miss for cache level 3
#hits
n_hits for cache level 1
n_hits for cache level 2
n_hits for cache level 3
config_3_level_cache_0.5x_assoc
config_3_level_cache_base
config_3_level_cache_2x_assoc


Miss Rate
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
Hit Rate
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_assoc
config_3_level_cache_base
config_3_level_cache_2x_assoc

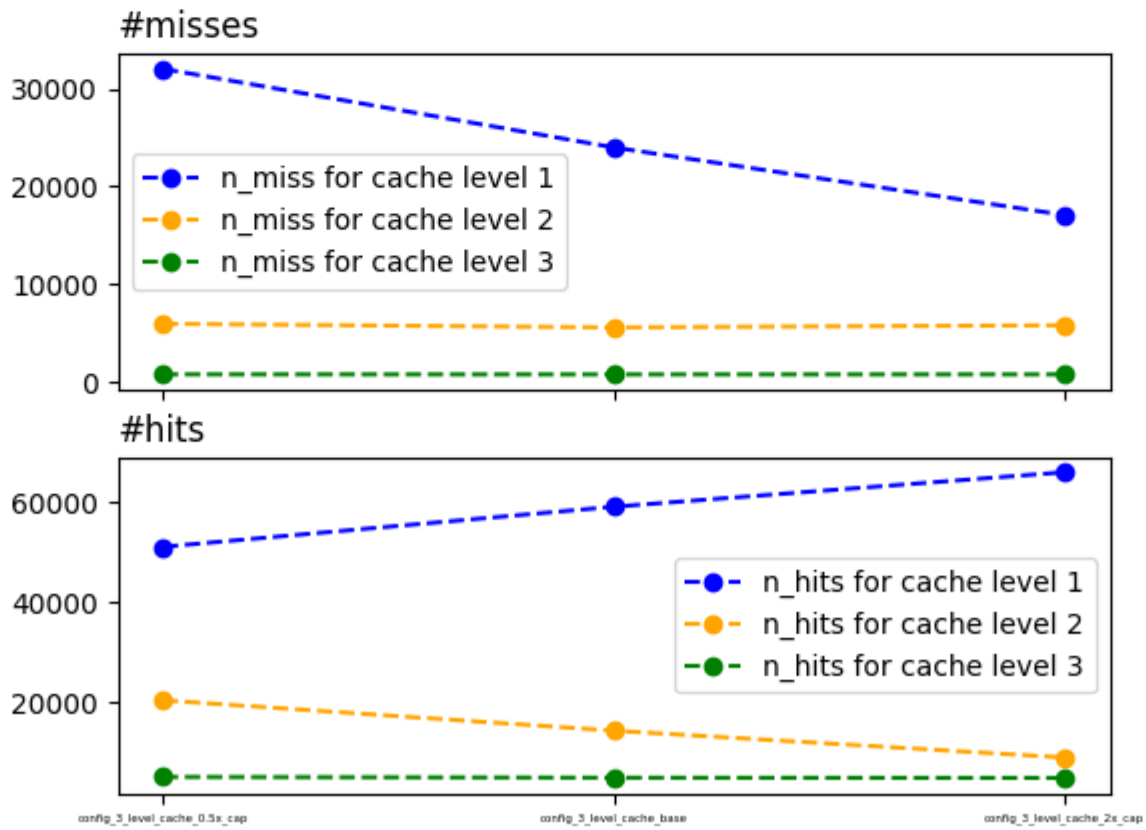


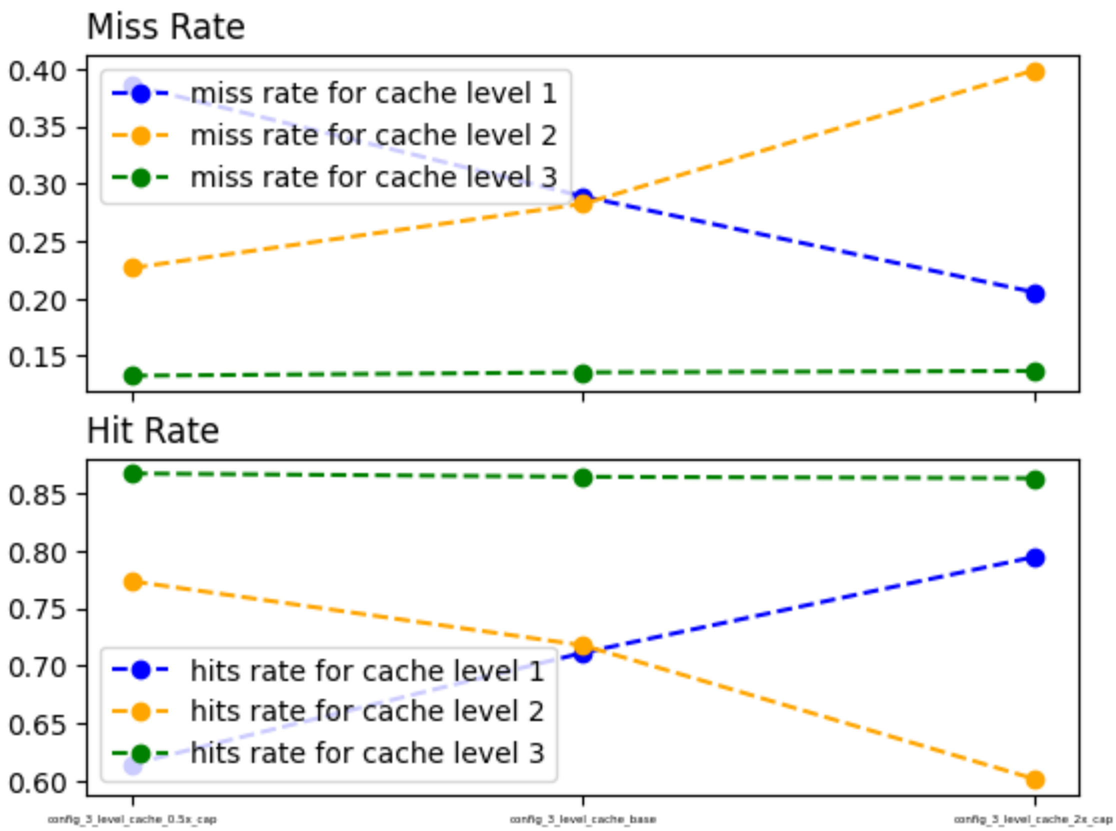
Experiments on Cache Capacity

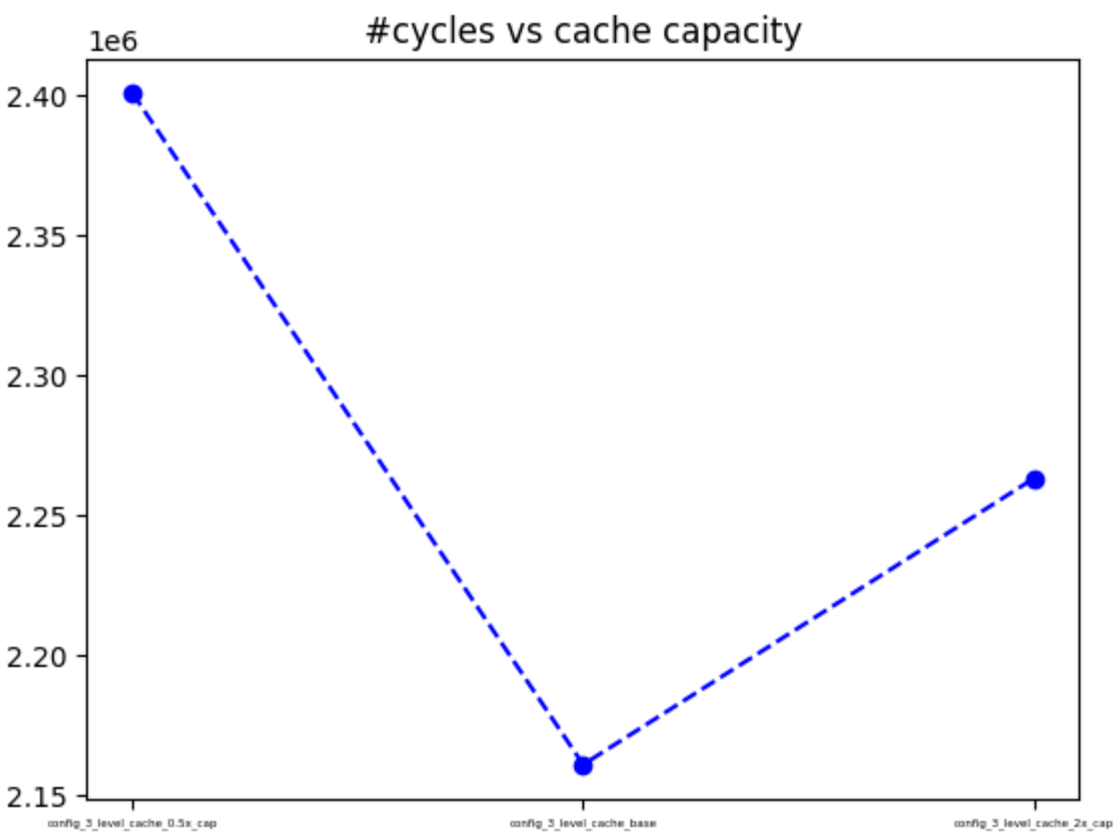
Here we take the 3 level cache as base and reduce the L1 cache capacity by half, and also increase the base's L1 cache capacity twice.

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_each_cache_data(axis[0], all_data["mat_transpose"], "n_miss", "#misses"
plot_each_cache_data(axis[1], all_data["mat_transpose"], "n_hits", "#hits",

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_rate(axis[0], all_data["mat_transpose"], "n_miss", "Miss Rate", experim
plot_rate(axis[1], all_data["mat_transpose"], "n_hits", "Hit Rate", experime

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_cycles(axis, all_data["mat_transpose"], "#cycles vs cache capacity", ex
```

```
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [32048, 23990, 17060] [5916, 5545, 5766] [734, 729, 731]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [51019, 59077, 66007] [20194, 14088, 8695] [4798, 4648,
4611]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [0.38580904571008945, 0.2888030144341315, 0.205376382919
81172] [0.226579854461892, 0.2824326389242602, 0.39872761219832653] [0.1326
825741142444, 0.13557745954993491, 0.1368401347809809]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [0.6141909542899106, 0.7111969855658685, 0.7946236170801
882] [0.773420145538108, 0.7175673610757398, 0.6012723878016735] [0.8673174
258857556, 0.8644225404500651, 0.8631598652190191]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [2401015, 2161079, 2263359]
```




Miss Rate
0.40
0.35
0.30
0.25
0.20
0.15
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
Hit Rate
0.85
0.80
0.75
0.70
0.65
0.60
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_cap
config_3_level_cache_base
config_3_level_cache_2x_cap

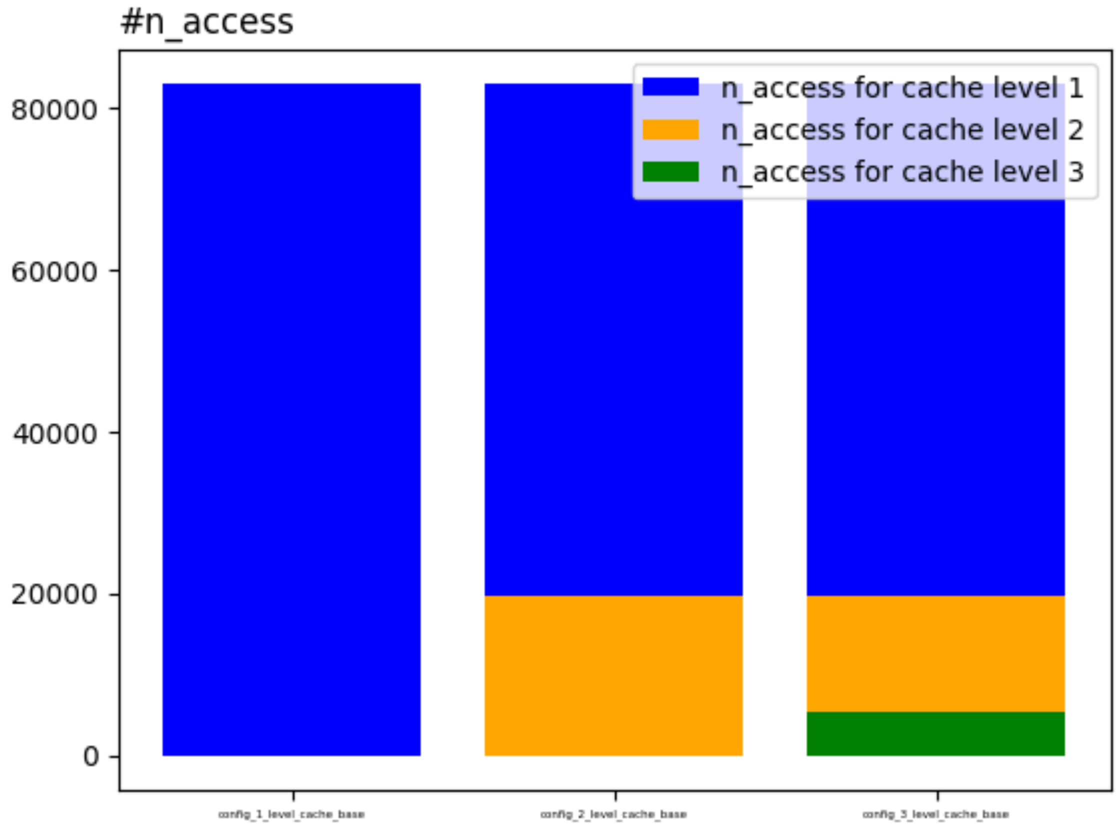

#cycles vs cache capacity
1e6
2.40
2.35
2.30
2.25
2.20
2.15
config_3_level_cache_0.5x_cap
config_3_level_cache_base
config_3_level_cache_2x_cap

Experiments on number of cache layers

Here we consider 3 level cache, 2 level cache and 1 level cache and plot amat vs cache layer graph

In [ ]:
```
figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_1_level_cache_base", "config_2_level_cache_base", "co
plot_amat(axis, all_data["mat_transpose"], "n_access", "#n_access", experime
```

```
['config_1_level_cache_base', 'config_2_level_cache_base', 'config_3_level_
cache_base'] [83067, 83067, 83067] [-1, 19633, 19633] [-1, -1, 5377]
```



We observe the following for transpose kernel after playing around with the cache parameters

- Increasing block size from 8 to 32.
  - Increases #hits, hit rate for almost all caches
  - Decreases #cycles
- Increasing associativity from 1 to 4
  - Reduces #misses, miss rate for L1 cache
  - Decreases #cycles
- Increasing cache capacity from 8 to 32
  - Increases #hits, hit rates
  - Decreases #cycles and then increases

Cache Study - Column wise Matrix Copy Kernel

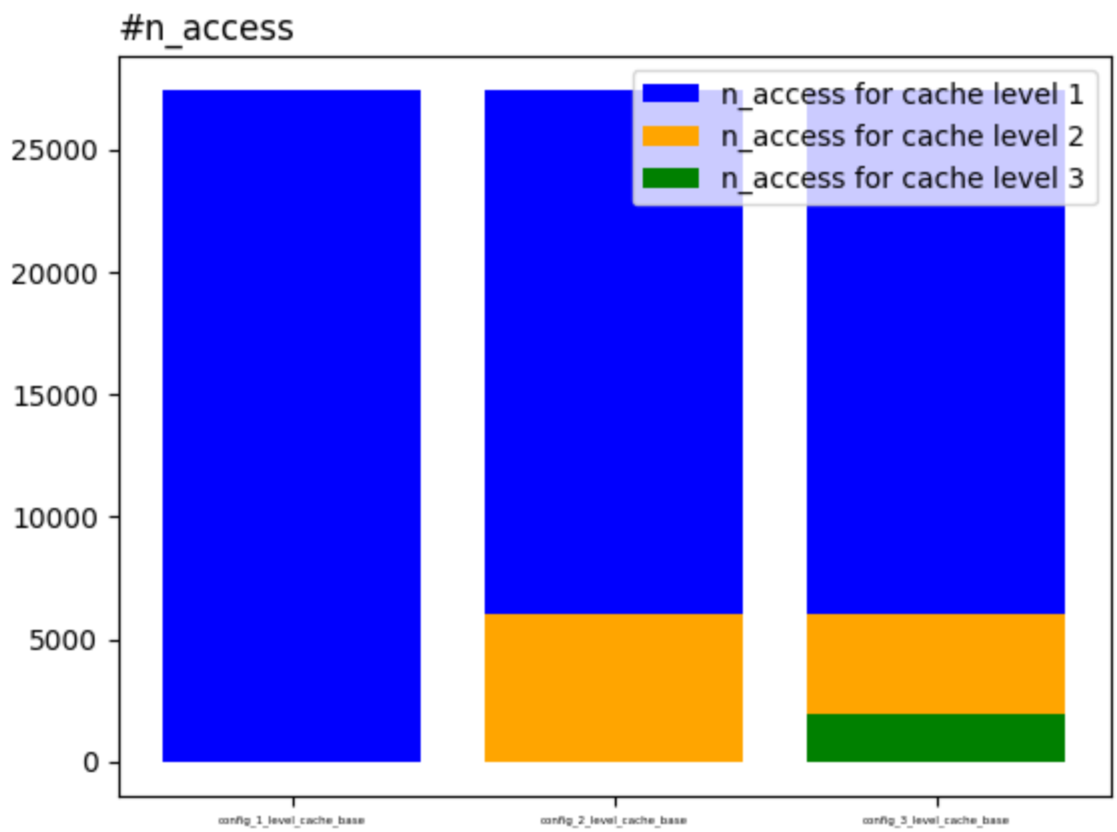
Column wise matrix copy operation reads data column by column and then writes the entry to a new matrix. We theorize that this implementation, having minimal spatial and temporal locality will not be cache friendly.

Experiments on number of cache layers

Here we consider 3 level cache, 2 level cache and 1 level cache and plot amat vs cache layer graph

In [ ]:
```
figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_1_level_cache_base", "config_2_level_cache_base", "co
plot_amat(axis, all_data["mat_column_wise_copy"], "n_access", "#n_access", e
```

```
['config_1_level_cache_base', 'config_2_level_cache_base', 'config_3_level_
cache_base'] [27458, 27458, 27458] [-1, 5997, 5997] [-1, -1, 1982]
```

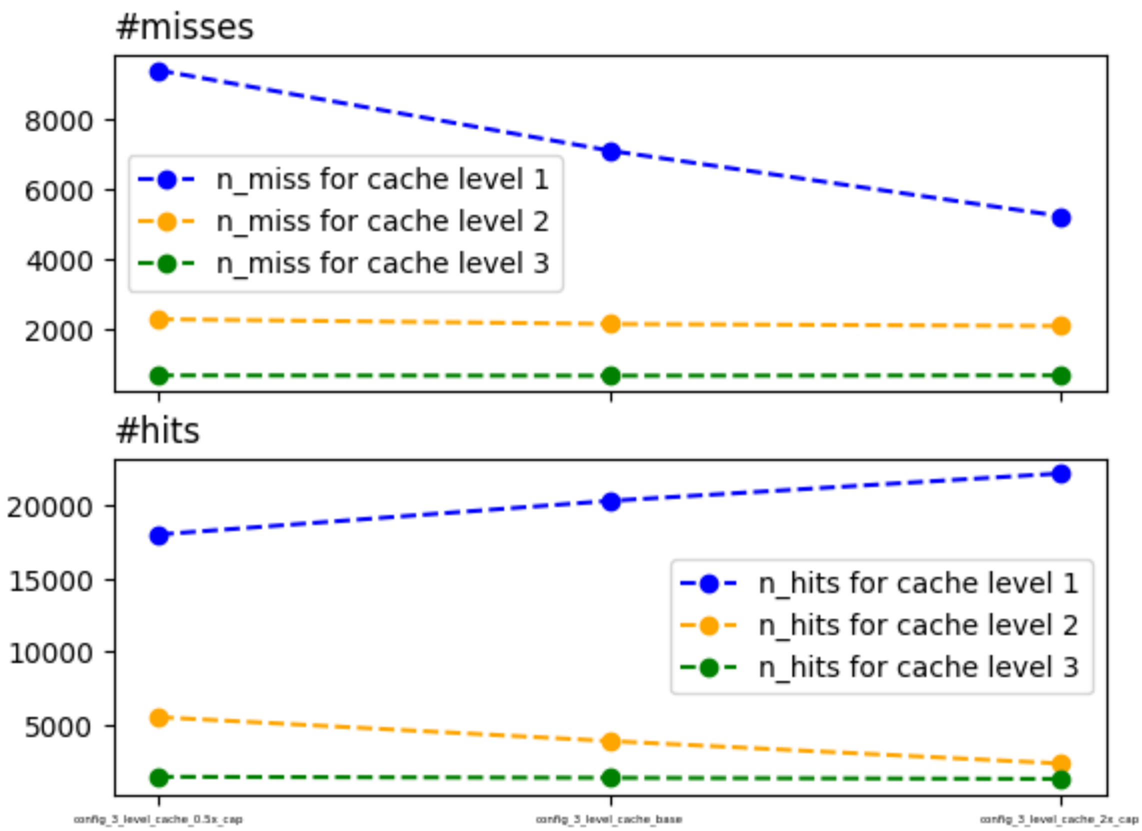


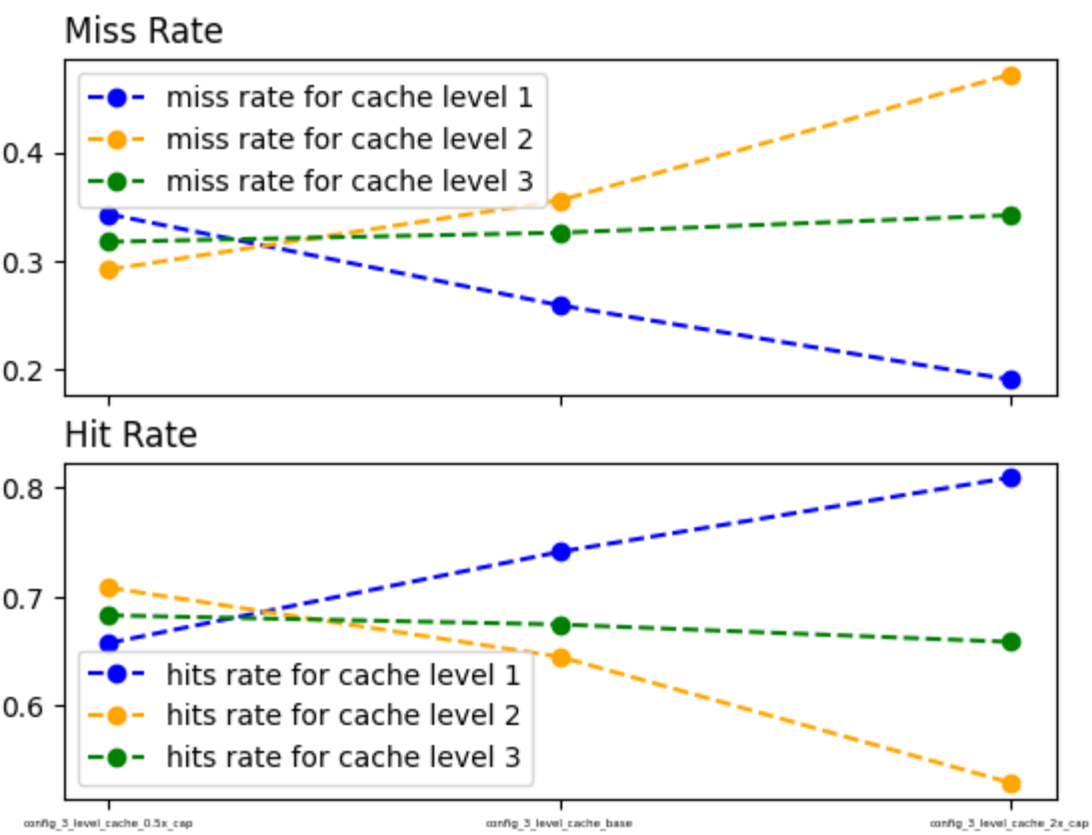
Experiments on Cache Capacity

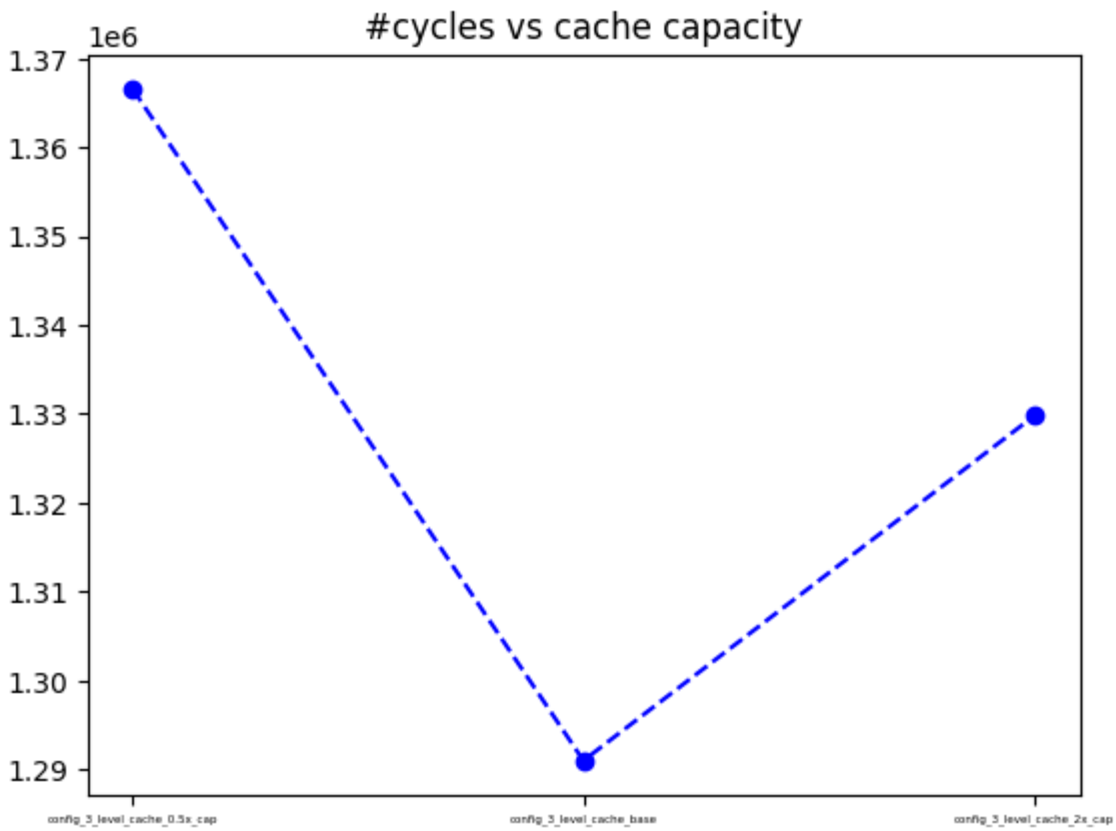
Here we take the 3 level cache as base and reduce the L1 cache capacity by half, and also increase the base's L1 cache capacity twice.

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_each_cache_data(axis[0], all_data["mat_column_wise_copy"], "n_miss", "#
plot_each_cache_data(axis[1], all_data["mat_column_wise_copy"], "n_hits", "#

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_rate(axis[0], all_data["mat_column_wise_copy"], "n_miss", "Miss Rate",
plot_rate(axis[1], all_data["mat_column_wise_copy"], "n_hits", "Hit Rate", e

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_cap", "config_3_level_cache_base",
plot_cycles(axis, all_data["mat_column_wise_copy"], "#cycles vs cache capaci
```

```
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [9420, 7112, 5230] [2270, 2133, 2075] [655, 646, 656]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [18038, 20346, 22228] [5504, 3864, 2322] [1408, 1336, 12
62]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [0.3430694151067084, 0.25901376647971447, 0.190472721975
3806] [0.29199897092873683, 0.35567783891945975, 0.47191266772799634] [0.31
74987881725642, 0.32593340060544906, 0.3420229405630866]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [0.6569305848932916, 0.7409862335202855, 0.8095272780246
194] [0.7080010290712632, 0.6443221610805403, 0.5280873322720037] [0.682501
2118274357, 0.6740665993945509, 0.6579770594369134]
['config_3_level_cache_0.5x_cap', 'config_3_level_cache_base', 'config_3_le
vel_cache_2x_cap'] [1366602, 1290974, 1329798]
```


#misses
8000
6000
4000
2000
n_miss for cache level 1
n_miss for cache level 2
n_miss for cache level 3
#hits
20000
15000
10000
5000
n_hits for cache level 1
n_hits for cache level 2
n_hits for cache level 3
config_3_level_cache_0.5x_cap
config_3_level_cache_base
config_3_level_cache_2x_cap


Miss Rate
0.4
0.3
0.2
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
Hit Rate
0.8
0.7
0.6
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_cap
config_3_level_cache_base
config_3_level_cache_2x_cap

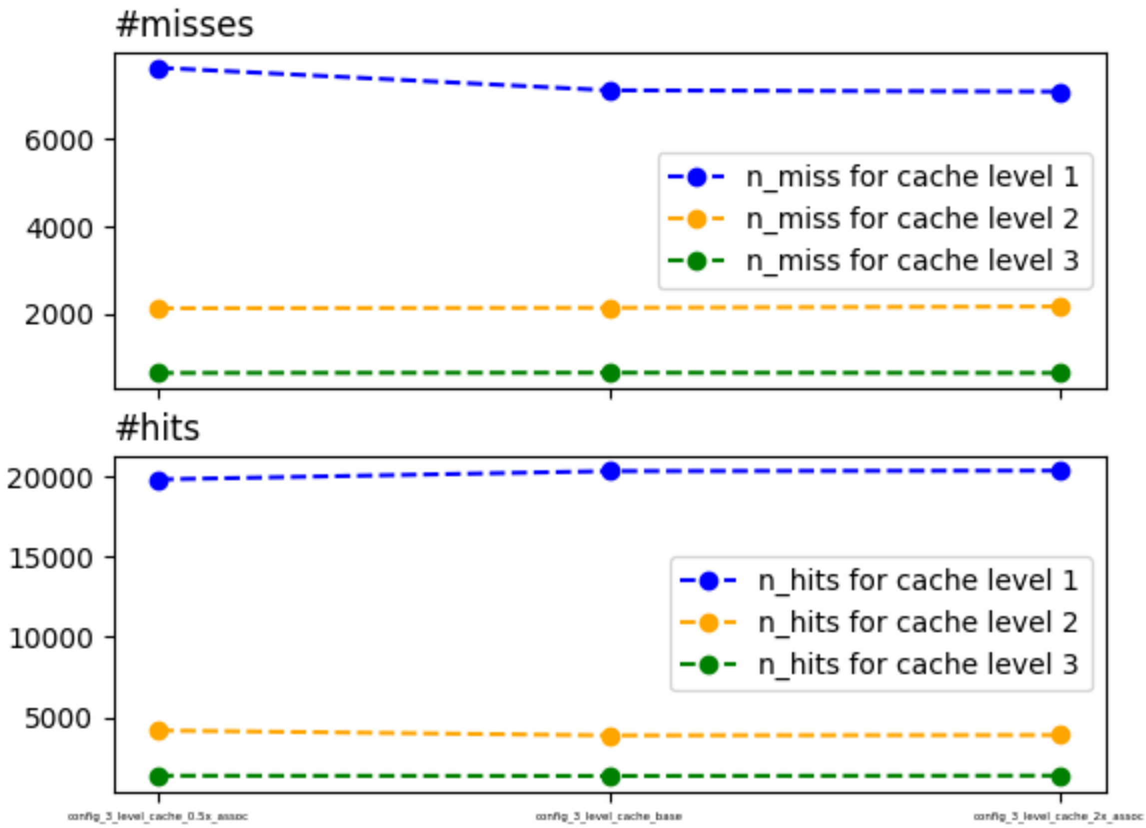


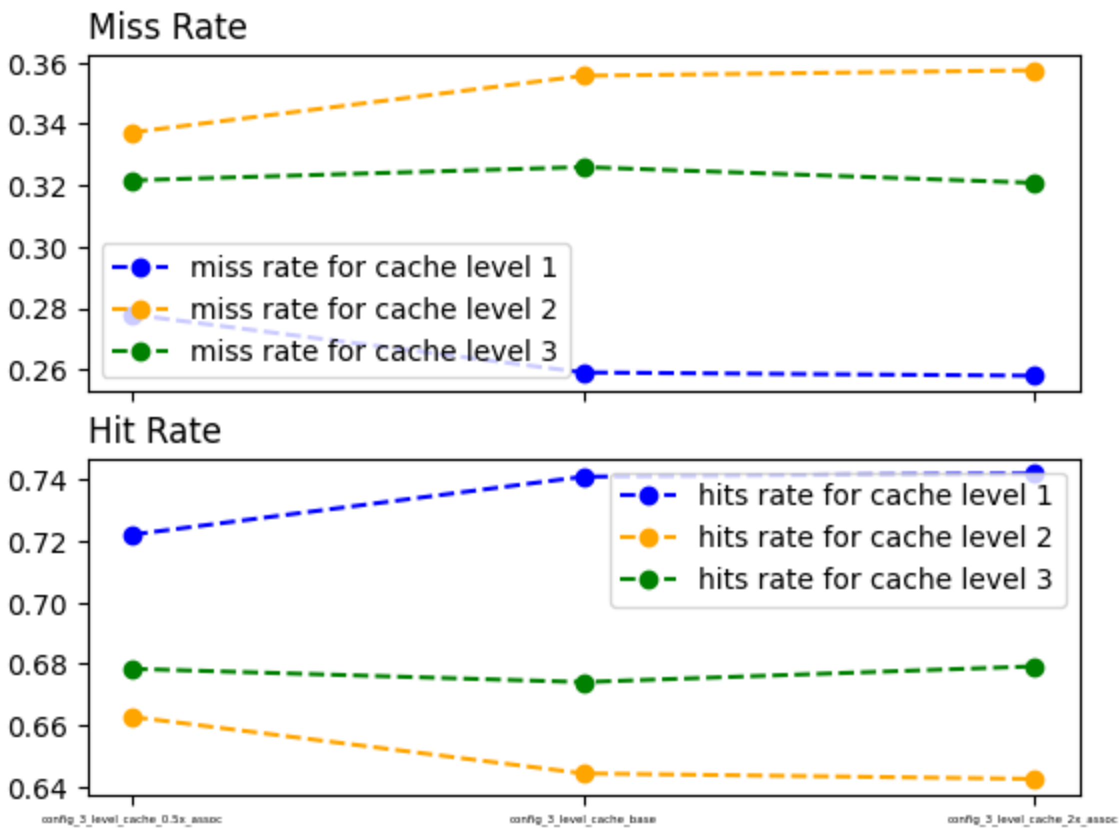
Experiments on Cache Associativity

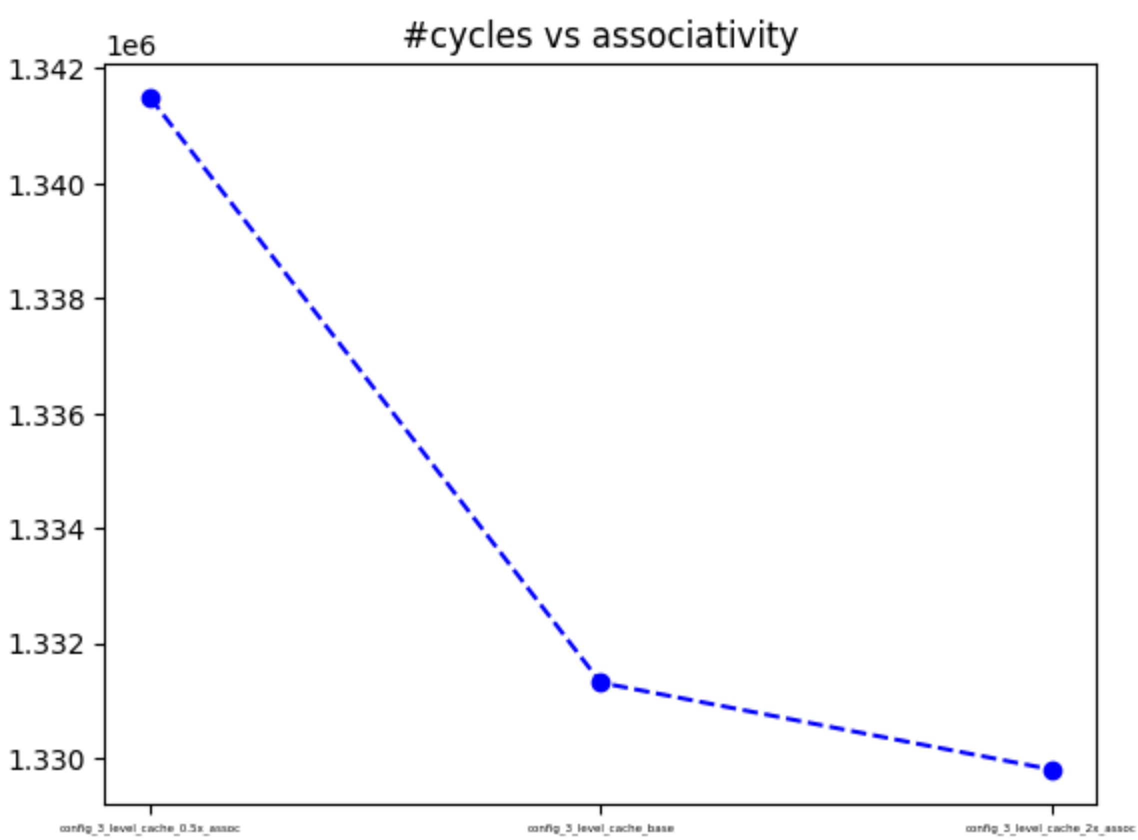
Here we take the 3 level cache as base and reduce the associativity of L1 cache size by half, and also increase the base's L1 cache associativity twice.

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_each_cache_data(axis[0], all_data["mat_column_wise_copy"], "n_miss", "#
plot_each_cache_data(axis[1], all_data["mat_column_wise_copy"], "n_hits", "#

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_rate(axis[0], all_data["mat_column_wise_copy"], "n_miss", "Miss Rate",
plot_rate(axis[1], all_data["mat_column_wise_copy"], "n_hits", "Hit Rate", e

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_assoc", "config_3_level_cache_base
plot_cycles(axis, all_data["mat_column_wise_copy"], "#cycles vs associativit
```

```
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [7630, 7112, 7083] [2122, 2133, 2162] [640, 646, 64
0]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [19828, 20346, 20375] [4170, 3864, 3886] [1350, 133
6, 1355]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [0.27787894238473304, 0.25901376647971447, 0.2579576
0798310147] [0.33725365543547364, 0.35567783891945975, 0.357473544973545]
[0.32160804020100503, 0.32593340060544906, 0.3208020050125313]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [0.7221210576152669, 0.7409862335202855, 0.742042392
0168986] [0.6627463445645264, 0.6443221610805403, 0.642526455026455] [0.678
391959798995, 0.6740665993945509, 0.6791979949874687]
['config_3_level_cache_0.5x_assoc', 'config_3_level_cache_base', 'config_3_
level_cache_2x_assoc'] [1341498, 1331310, 1329798]
```




Miss Rate
0.36
0.34
0.32
0.30
0.28
0.26
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
Hit Rate
0.74
0.72
0.70
0.68
0.66
0.64
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_assoc
config_3_level_cache_base
config_3_level_cache_2x_assoc

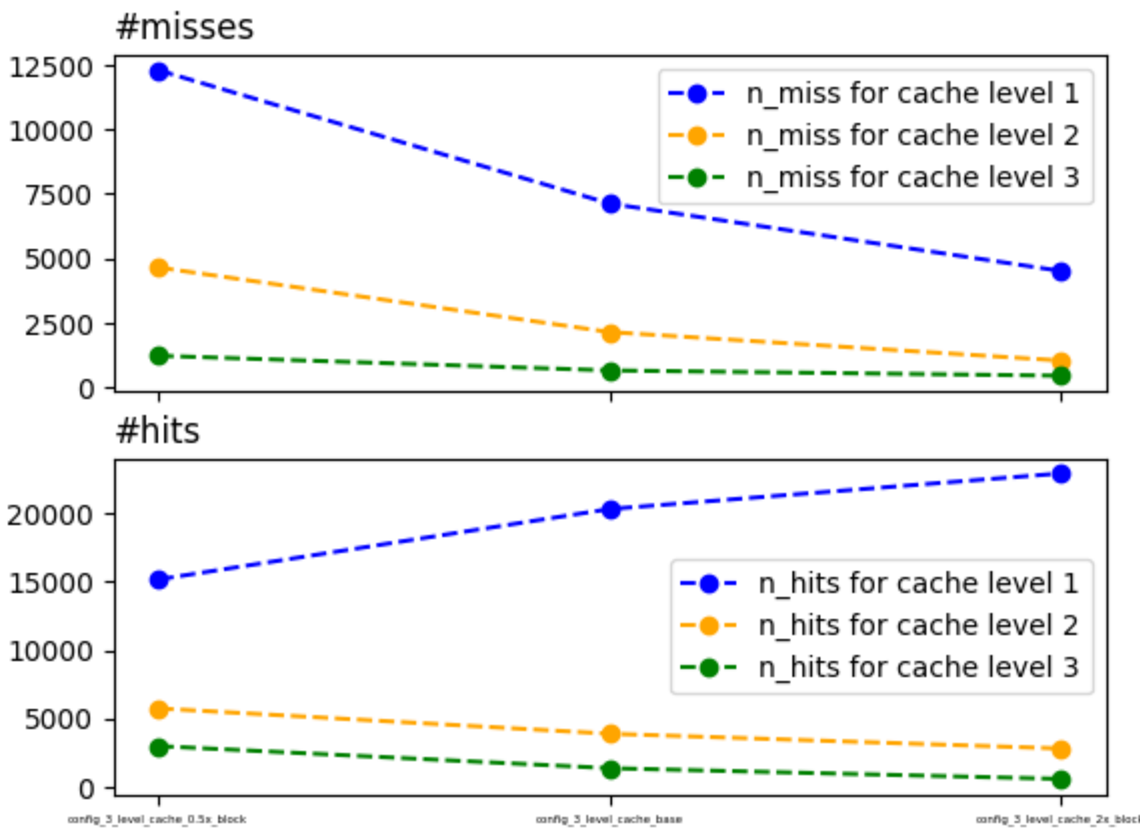

#cycles vs associativity
1e6
1.342
1.340
1.338
1.336
1.334
1.332
1.330
config_3_level_cache_0.5x_assoc
config_3_level_cache_base
config_3_level_cache_2x_assoc

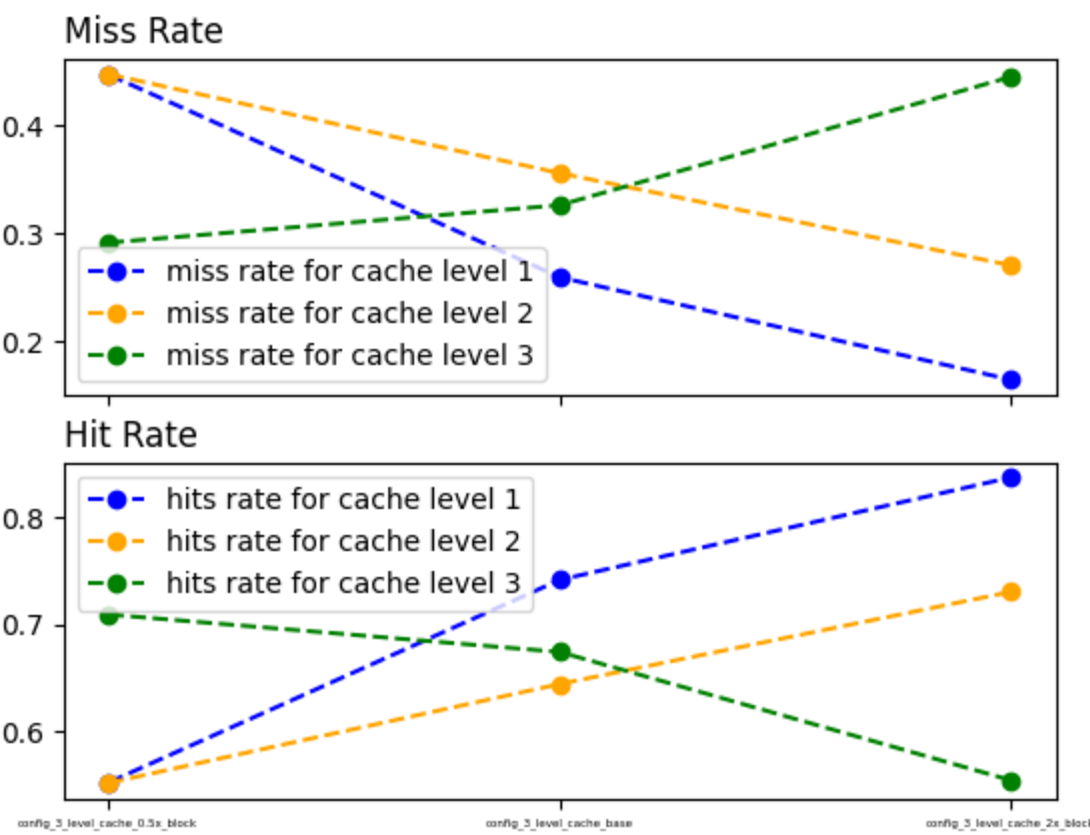
Experiments on Block size

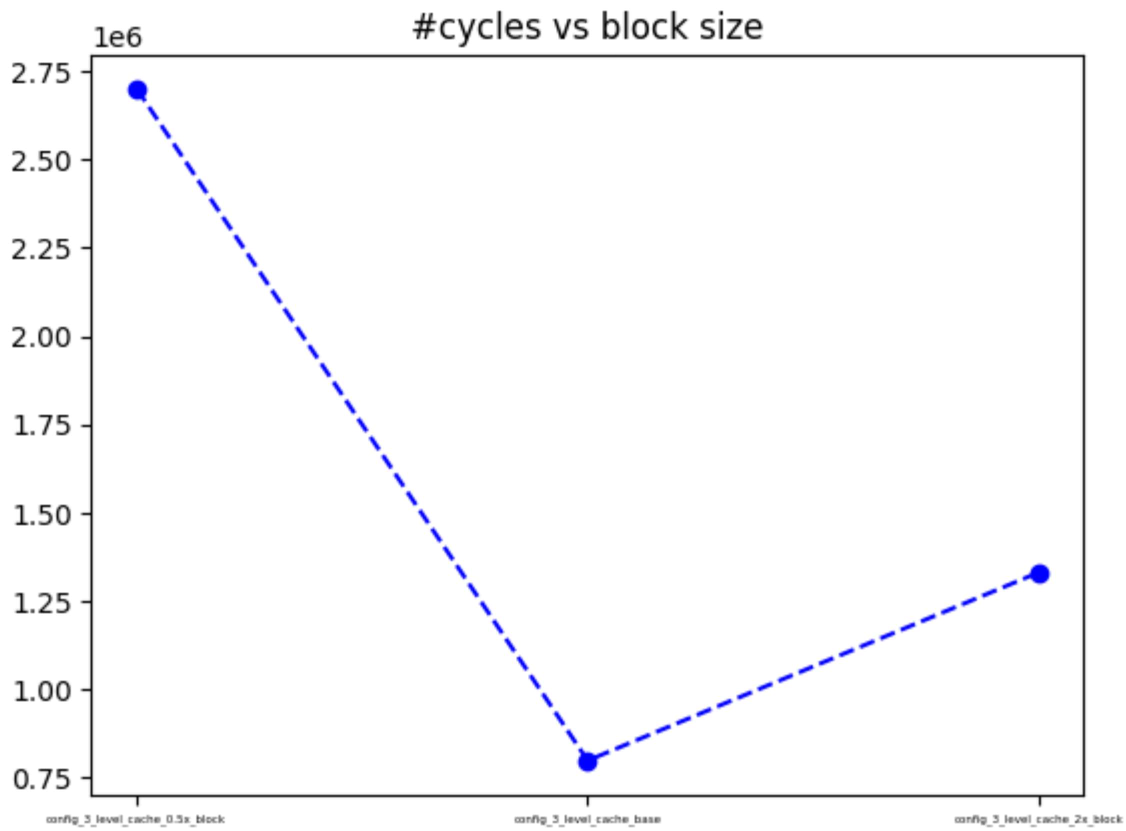
Here we take the 3 level cache as base and reduce its block size by half, and also increase the base's block size twice.

In [ ]:
```
figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_each_cache_data(axis[0], all_data["mat_column_wise_copy"], "n_miss", "#
plot_each_cache_data(axis[1], all_data["mat_column_wise_copy"], "n_hits", "#

figure, axis = plt.subplots(2, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_rate(axis[0], all_data["mat_column_wise_copy"], "n_miss", "Miss Rate",
plot_rate(axis[1], all_data["mat_column_wise_copy"], "n_hits", "Hit Rate", e

figure, axis = plt.subplots(1, sharex=True,)
experiments = ["config_3_level_cache_0.5x_block", "config_3_level_cache_base
plot_cycles(axis, all_data["mat_column_wise_copy"], "#cycles vs block size",
```

```
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [12277, 7112, 4501] [4634, 2133, 1029] [1215, 646, 4
41]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [15181, 20346, 22957] [5722, 3864, 2780] [2958, 133
6, 550]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [0.44711923665234177, 0.25901376647971447, 0.1639230
8252603977] [0.4474700656624179, 0.35567783891945975, 0.27014964557626675]
[0.291157440690151, 0.32593340060544906, 0.4450050454086781]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [0.5528807633476582, 0.7409862335202855, 0.836076917
4739602] [0.5525299343375821, 0.6443221610805403, 0.7298503544237332] [0.70
88425593098491, 0.6740665993945509, 0.5549949545913219]
['config_3_level_cache_0.5x_block', 'config_3_level_cache_base', 'config_3_
level_cache_2x_block'] [2701426, 796362, 1329798]
```


#misses
12500
10000
7500
5000
2500
0
n_miss for cache level 1
n_miss for cache level 2
n_miss for cache level 3
#hits
20000
15000
10000
5000
0
n_hits for cache level 1
n_hits for cache level 2
n_hits for cache level 3
config_3_level_cache_0.5x_block
config_3_level_cache_base
config_3_level_cache_2x_block


Miss Rate
0.4
0.3
0.2
miss rate for cache level 1
miss rate for cache level 2
miss rate for cache level 3
Hit Rate
0.8
0.7
0.6
hits rate for cache level 1
hits rate for cache level 2
hits rate for cache level 3
config_3_level_cache_0.5x_block
config_3_level_cache_base
config_3_level_cache_2x_block



We observe the following for column wise copy kernel after playing around with the cache parameters

- Increasing block size from 8 to 32.
  - Increases #hits, hit rate for almost all caches
  - Decreases #cycles and then increases
- Increasing associativity from 1 to 4
  - Reduces #misses, miss rate for L1 cache
  - Decreases #cycles
- Increasing cache capacity from 8 to 32
  - Increases #hits, hit rates
  - Decreases #cycles and then increases

Conclusion

As part of this exercise, we are able to understand how the architecture of cache systems impact the implementations of popular matrix operations. Although the current kernel codes analyzed as part of this assignment use naive implementation, we list down few possible optimizations here in order to work in future

- Block based matrix transpose
  - Do matrix transpose on the input matrix a block level instead of matrix level.
  - This approach will use both temporal and spatial locality to the best
- Scatter and Gather implementation using CSR matrix format
  - With increasing sparsity of the matrix, the CSR format based implementation would be a cache friendly algorithm

References

- [1] H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun and A. Karunanidhi, Pinpointing Representative Portions of Large Intel ® Itanium ® Programs with Dynamic Instrumentation 37th International Symposium on Microarchitecture (MICRO-37'04), 2004, pp. 81-92, doi: 10.1109/MICRO.2004.28.
- [2] CacheSimulator, https://github.com/abhishekk06/CachePerformanceOnMatMul
- [3] GEM5 building, https://www.gem5.org/documentation/general_docs/building