Analysis of Performance Data

2023.12.05 Mon

Jeongin Park, Seoul National University
Max Plank Computing and Data Facility
2023.09.11 ~ 2023.12.04

MAX PLANCK
COMPUTING & DATA FACILITY
MPCDF

Tables of Content

Project Goal

- Measurement In addition to the benefits for the student in terms of learning and experience, we'd aim towards useful production outcomes, for example:

  - **identification of 'good' or 'bad' jobs,** i.e. improving the 'black sheep' functionality in Splunk, alerting about 'bad' jobs in 'near time’

  - **anomaly detection**, related to 'good' and 'bad' jobs, identifying performance drops for jobs that had better performance previously

  - **identification of the bottleneck for specific jobs and applications**: compute, memory bandwidth, io, network, etc., helping to identify the actual requirements of our application mix for (future) HPC systems

  - **fingerprinting of unknown applications**, e.g. shedding light at the famous 'a.out' or 'python' mystery

Information about Data

- Two types of data: Job-Summary, Time-resolved data
- Job summary: collected after job ended

AI,AVG_LOAD,AVG_O_PTMP,AVG_O_U,AVG_R_NET,AVG_R_PTMP,**AVG_R_U,**AVG_W_NET,AVG_W_PTMP,AVG_W_U,FP_SCALAR,FP_VECTOR,GF,HPCMD_CHECKPOINT,MEM_PEAK,PEAK_LOAD,**PEAK_O_PTMP,**PEAK_O_U,PEAK_R_NET,PEAK_R_PTMP,PEAK_R_U,PEAK_W_NET,PEAK_W_PTMP,PEAK_W_U,SOCKET_BALANCE,VEC_RATIO,**cores,**elapsed,exe,exit_code,groupid,host,iter,jobend,jobid,jobstart,**maxrss,**min_empty_cores,njobsteps,nnodes,partition,timelimit,userid,datetime,timestamp

- Time-resolved data: collected every 4 min. From each nodes, sockets, I/Os, GPU, etc.

AI,ALGO_INT,AVG_LOAD,AVG_O_PTMP,AVG_O_U,AVG_R_NET,AVG_R_PTMP,AVG_R_U,AVG_W_NET,AVG_W_PTMP,AVG_W_U,BR_MISS_RATIO,CACHE_MISS_RATIO,FORT_BUFFERED,FP_SCALAR,FP_VECTOR,GF,GFLOPS,HPCMD_CHECKPOINT,IPC,I_MPI_EXTRA_FILESYSTEM,I_MPI_HYDRA_BOOTSTRAP,I_MPI_LINK,I_MPI_PMI_LIBRARY,I_MPI_ROOT,LOADEDMODULES,MEM_BW,MEM_PEAK,MKLROOT,MKL_DOC,MKL_HOME,MODULESHOME,NodeRSS,OMP_NUM_THREADS,OMP_STACKSIZE,PEAK_LOAD,PEAK_O_PTMP,PEAK_O_U,PEAK_R_NET,PEAK_R_PTMP,PEAK_R_U,PEAK_W_NET,PEAK_W_PTMP,PEAK_W_U,PMI_FD,PMI_JOBID,PMI_RANK,PMI_SIZE,PWD,PYTHONPATH,PYTHONSTARTUP,SLURM_CELL,SLURM_CLUSTER_NAME,SLURM_CONF,SLURM_CPUS_ON_NODE,SLURM_CPU_BIND_TYPE,SLURM_CPU_BIND_VERBOSE,SLURM_DISTRIBUTION,SLURM_EXPORT_ENV,SLURM_GET_USER_ENV,SLURM_GTIDS,SLURM_JOBID,SLURM_JOB_ACCOUNT,SLURM_JOB_CPUS_PER_NODE,SLURM_JOB_CPUS_PER_NODE_PACK_GROUP_0,SLURM_JOB_GID,SLURM_JOB_ID,SLURM_JOB_NAME,SLURM_JOB_NODELIST,SLURM_JOB_NUM_NODES,SLURM_JOB_PARTITION,SOCKET_BALANCE,VEC_RATIO,awake,branch_misses,branches,cache_misses,cache_references,cores,cpu,cycles,elapsed,empty_cores,environment,epoch,exe,exit_code,fp_128d,fp_128s,fp_256d,fp_256s,fp_512d,fp_512s,fp_d,fp_s,fsid,gpfs_bytes_read,gpfs_bytes_written,gpfs_closes,gpfs_inode_updates,gpfs_opens,gpfs_reads,gpfs_writes,groupid,host,instructions,iter,jobend,jobid,jobname,jobstart,lib,load_15min,load_1min,load_5min,loadedmodules,major_faults,maxrss,min_empty_cores,minor_faults,njobsteps,nnodes,nodeid,ntasks,ntasks_per_node,omp_num_threads,opmode,partition,path,realmemory,rx_bytes,rx_packets,sockets,threadspercore,timelimit,total_threads,tx_bytes,tx_packets,type,userid,_time

Pre-processing data

- Raw Data
    - Several lines for each timestamp

```
,AI,ALGO_INT,AVG_LOAD,AVG_O_PTMP,AVG_O_U,AVG_R_NET,AVG_R_PTMP,AVG_R_U,AVG_W_NET,AVG_W_PTMP,AVG_W_U,BR_MISS_RATIO,CACHE_MISS_RATIO,FORT_BUFF
0,,,,,,,,,,,,,,,,,,thread,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,S1,,,0.0,,,,,,,,,,,,,,,,,,,,,,ravc2421,,,,7229318,,,
1,,,,,,,,,,,,,,,,,,thread,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,S0,,,0.0,,,,,,,,,,,,,,,,,,,,,,ravc2421,,,,7229318,,,
2,,,,,,,,,,,,,,,,,,exe,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,aims.x,,,,,,,,,,,,,,,,,,,,ravc2421,,,,7229318,,,,6
3,,,,,,,,,,,,,,,,,,thread,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,S1,,,0.0,,,,,,,,,,,,,,,,,,,,,,ravc2427,,,,7229318,,,
4,,,,,,,,,,,,,,,,,,thread,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,S0,,,0.0,,,,,,,,,,,,,,,,,,,,,,ravc2427,,,,7229318,,,
5,,,,,,,,,,,,,,,,,,exe,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,aims.x,,,,,,,,,,,,,,,,,,,,ravc2427,,,,7229318,,,,6
6,,,,,,,,,,,,,,,,,,network,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,ravc2420,,,,7229318,,,,,,,
7,,,,,,,,,,,,,,,,,,memory,,,,,,,,,,,,,,,82063.796875,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,S0,,,,,,,,,,,,,,,,,,,,,,,,,,,ravc2420,,,,7
8,,,,,,,,,,,,,,,,,,network,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,ravc2418,,,,7229318,,,,,,,
```

- What we want: One line for each timestamp, one .csv file for each job

Pre-processing data

- Several types of data for each timestamp and different columns for each type
  - 'thread' 'exe' 'network' 'memory' 'perf' 'gpfs' 'env' 'libs' ‘job_start', ‘job_summary’
  - different socket names, different node names for same timestamp

```
thread
Index(['Unnamed: 0', 'HPCMD_CHECKPOINT', 'cpu', 'empty_cores', 'host', 'jobid',
       'nodeid', 'total_threads', 'datetime', 'timestamp'],
      dtype='object')

perf
Index(['Unnamed: 0', 'ALGO_INT', 'BR_MISS_RATIO', 'CACHE_MISS_RATIO',
       'FP_SCALAR', 'FP_VECTOR', 'GFLOPS', 'HPCMD_CHECKPOINT', 'IPC', 'MEM_BW',
       'branch_misses', 'branches', 'cache_misses', 'cache_references', 'cpu',
       'cycles', 'fp_128d', 'fp_128s', 'fp_256d', 'fp_256s', 'fp_512d',
       'fp_512s', 'fp_d', 'fp_s', 'host', 'instructions', 'jobid',
...
       'iter', 'jobend', 'jobid', 'jobstart', 'maxrss', 'min_empty_cores',
       'njobsteps', 'nnodes', 'partition', 'timelimit', 'userid', 'datetime',
```

- ‘**node_name/<info>**’, ’**node_name/socket_name/<info>**’ or ‘**I/O_name/<info>**’ format

```
1  ,timestamp,ravc2418/load_15min,ravc2418/load_1min,ravc2418/load_5min,ravc2418/rx_bytes,ravc2418/rx_packets,ravc2418/tx_bytes,ravc2418/tx_p
2  0,1694628950.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,5.473893546583224,0.0014012103095447,0.4497905626243322,2621087287192.0,12037020592216.0,415.36
3  1,1694629190.0,57.82,72.25,61.69,214294647140.0,102624566.0,209813712560.0,101613163.0,5.459401095993919,0.0013827552937068,0.465849533443
4  2,1694629430.0,61.16,72.42,67.62,199017897388.0,94793970.0,194186666888.0,93822710.0,4.909434484389877,0.0013393595367307,0.45399112203268
5  3,1694629670.0,63.68,72.35,70.21,211600258352.0,100550868.0,207259884076.0,99634207.0,5.466917943818227,0.0013117692388565,0.4652687427942
6  4,1694629910.0,65.65,72.18,71.36,188031117184.0,89933819.0,183275663948.0,89023140.0,4.935401325329669,0.0013614445403118,0.43679370068457
7  5,1694630150.0,67.14,72.11,71.79,191073032524.0,90901077.0,186293767580.0,89969663.0,4.865335333927567,0.0012609366382344,0.45176921763055
8  6,1694630390.0,68.3,72.23,72.03,203172586760.0,96442786.0,198988172220.0,95639065.0,5.351997430564679,0.0013292961086549,0.4458934616314655
9  7,1694630630.0,69.23,72.49,72.2,200530469828.0,96098941.0,195488002960.0,95159326.0,5.299342361780027,0.0013724526159 48,0.4684977290460191
10 8,1694630870.0,69.96,72.37,72.32,198176399556.0,94176931.0,193296393040.0,93285542.0,4.898562655708133,0.0013315943169103,0.45206901910709
```

- Klaus used it to export data in 2d format

Pre-processing data

- For training, One Line per Job
  - Statistics
    - 'min', 'max', 'mean', 'std', 'skew', 'p_5th', 'p_25th', 'p_50th', 'p_75th', ‘p_90th'

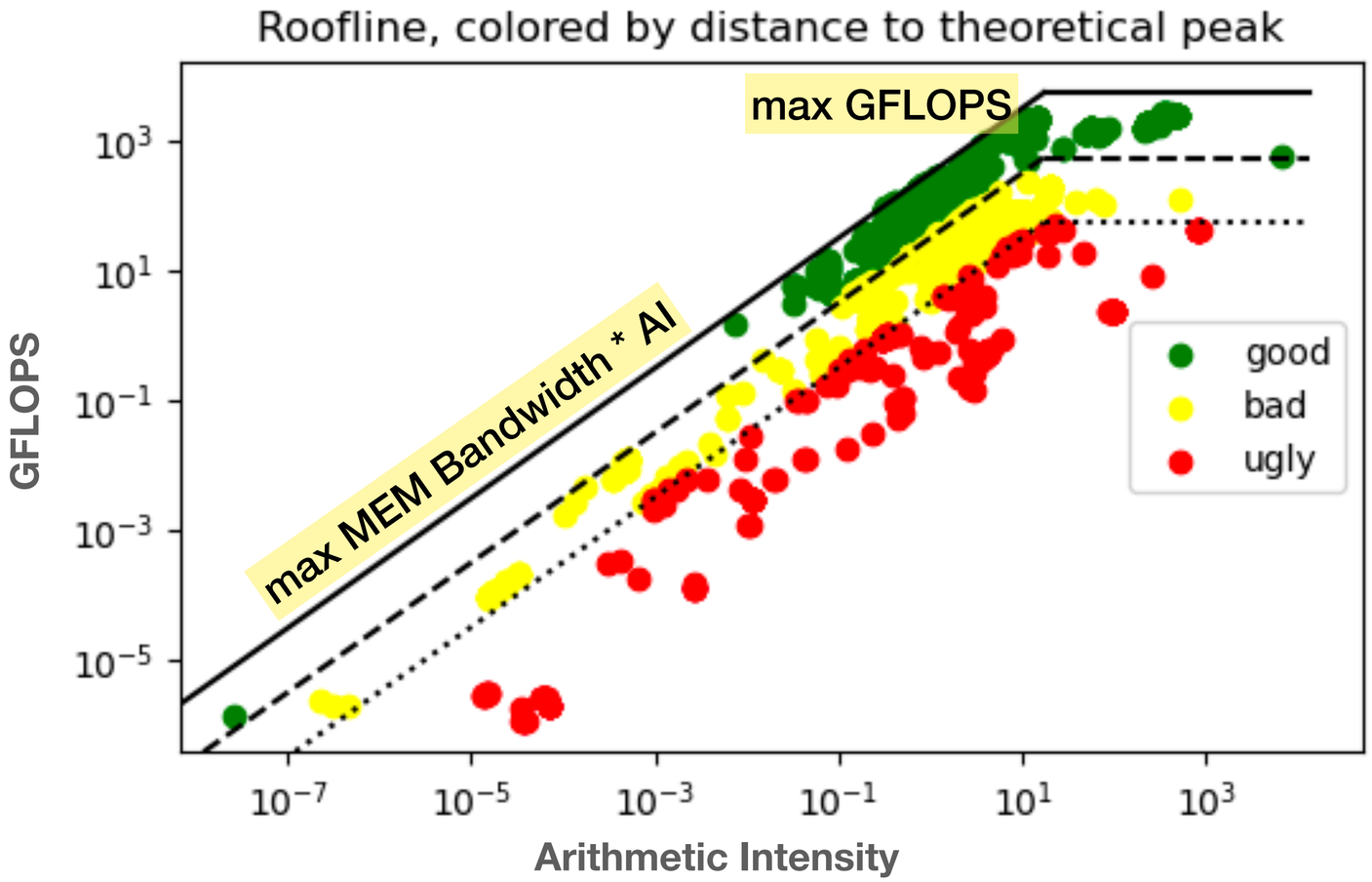
	ALGO_INT/min	ALGO_INT/max	ALGO_INT/mean	ALGO_INT/std	ALGO_INT/skew	ALGO_INT/p_5th	ALGO_INT/p_25th	ALGO_INT/p_50th	ALGO_INT/p_75th
5541946	1.8654233549940626…	9.786837608749712	1.7479334929069141	0.0		9.786837608749712	9.786837608749712	9.786837608749712	9.786837608749712
5541945	2.262394929717692e…	10.754192454609774	1.8153478291346221	0.0		10.754192454609774	10.754192454609774	10.754192454609774	10.754192454609774
5539854	7.0166040396117e-05	1.7035379952828469	0.5184217023777926	0.0		1.7035379952828469	1.7035379952828469	1.7035379952828469	1.7035379952828469

  - Windows

	0/ALGO_INT	0/BR_MISS_RATIO	0/CACHE_MISS_RAT…	0/FP_SCALAR	0/FP_VECTOR	0/GFLOPS	0/IPC	0/MEM_BW	0/NodeRSS
5541946	0.978895336507698	0.0001853966029124	0.19893198159274275	135617044971.0	32047140382.0	1.1462376219956523	3.06100972155864	1.1709501304653913	2140.22265625
5541945	0.9289668507382656	0.0001975268257705	0.2029060536260196	136712436258.5	37391755921.0	1.243824443923913	3.0682623160023295	1.3389330770365218	2141.859375
5539854	0.5051002730985245	0.00435354958236955	0.3655553629601649	16345416.5	1892561722764.5	32.87044422540653	0.9532457090208877	65.0770668243033	5344.92578125

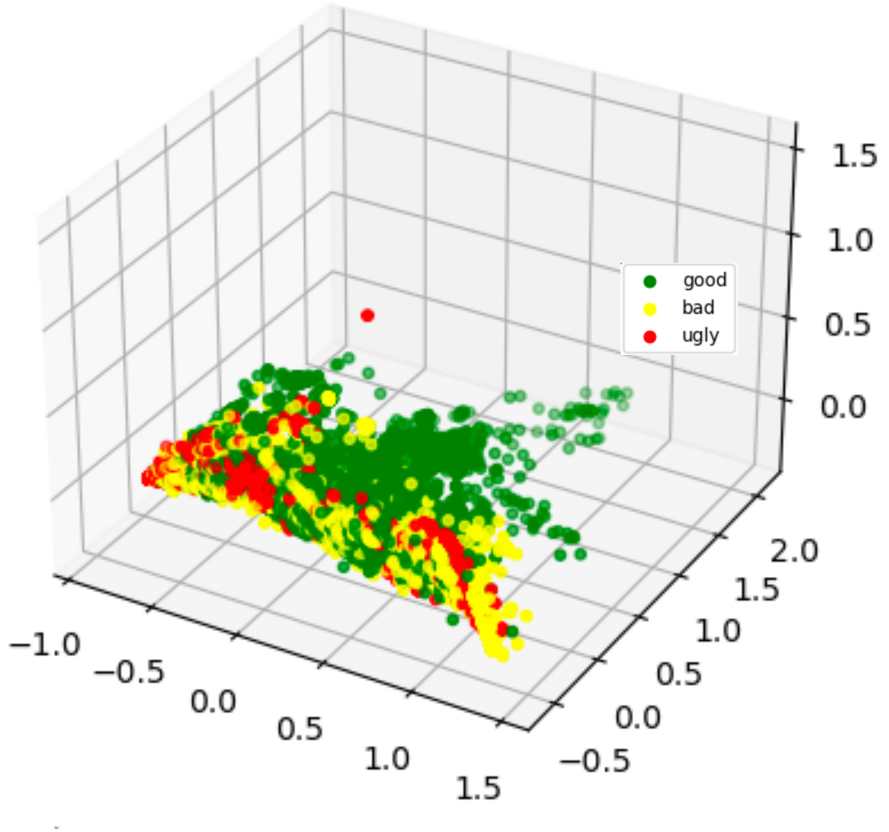
Roofline Model

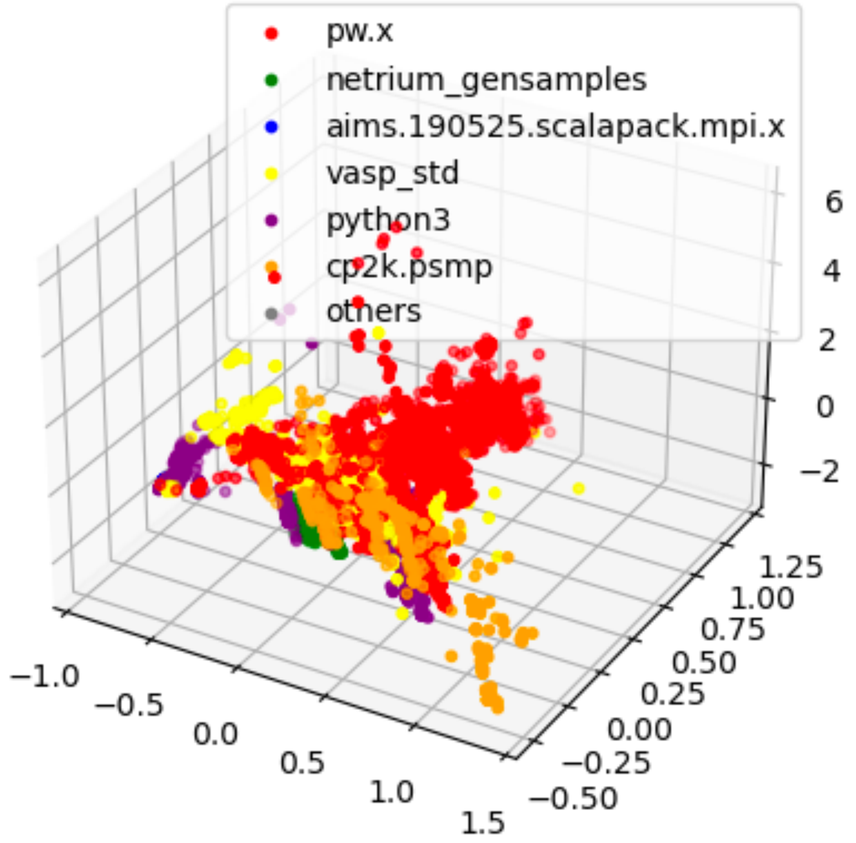
- Measurement of Computer Performance
- Can be calculated with Job-Summary data
- Two types of ceiling
  - One by memory bandwidth
  - Other by processor's performance
- Classified jobs into three: Good, Bad, Ugly



Several Methods we tried…

PCA





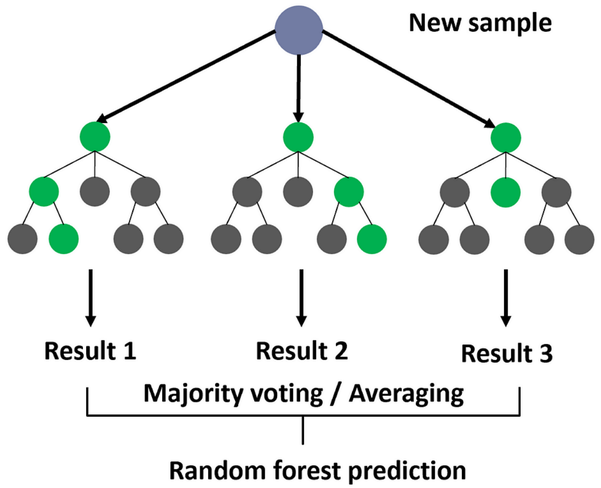
Why Random Forest

- PCA: just perturbing the data dimension. De-correlate correlated data and get new features
  - We cannot see which feature is important and which is not
  - Though we can see that it differentiates different Executables
- Random Forest: preserve features, methods to see compare feature importance

What is Random Forest

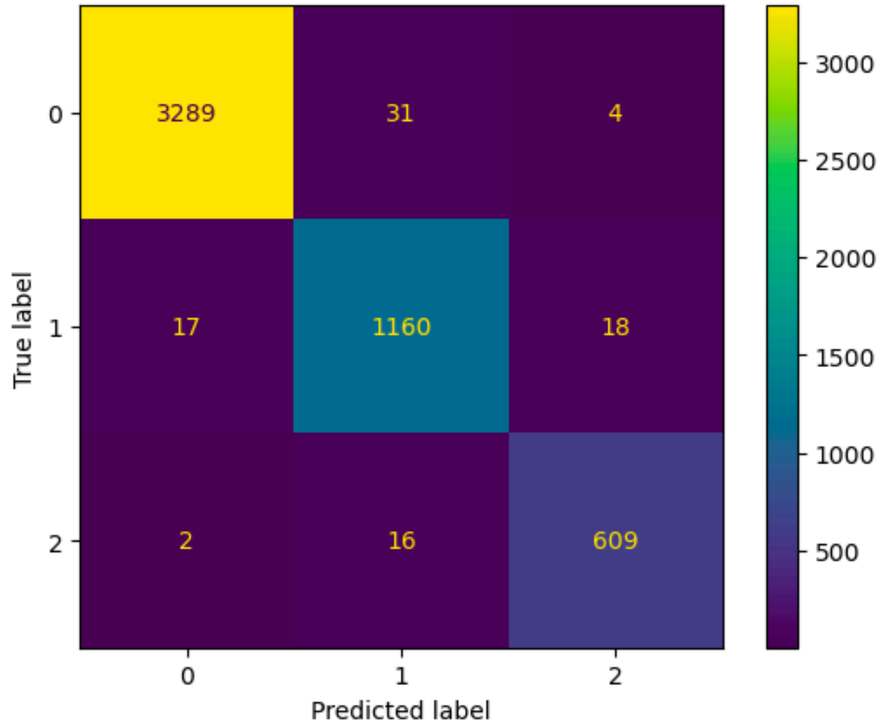
Ensemble of **Decision Trees**

- Decision Tree: used in classification problem
- Ensemble: made up of a set of classifiers—e.g. decision trees—and their predictions are aggregated to identify the most popular result
  - Randomly select some data, Train several different trees independently, Aggregate results



Training of Random Forest

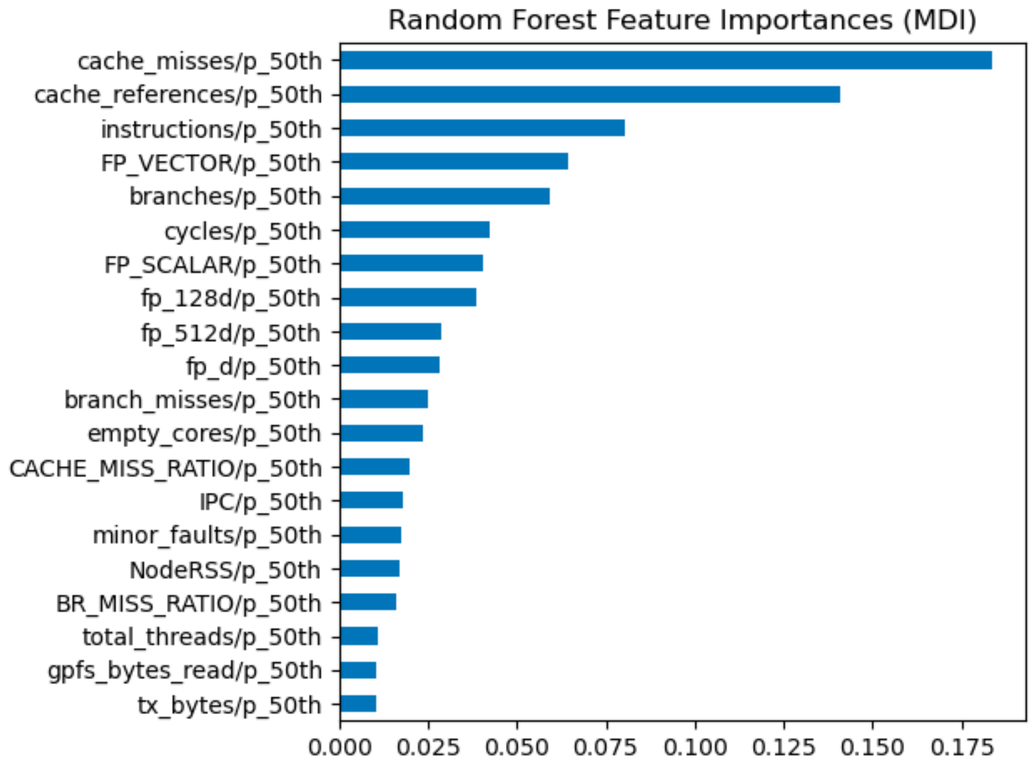
- Packages used: **`sklearn.ensemble`**.RandomForestClassifier
- Things to Consider
  - unbalanced dataset —> SMOTE to balance dataset
    - **Good** vs. Bad vs. Ugly
    - **Memory-bounded** vs. Compute-bounded
  - CPU vs. GPU: different columns —> Train separate model
- Test Result

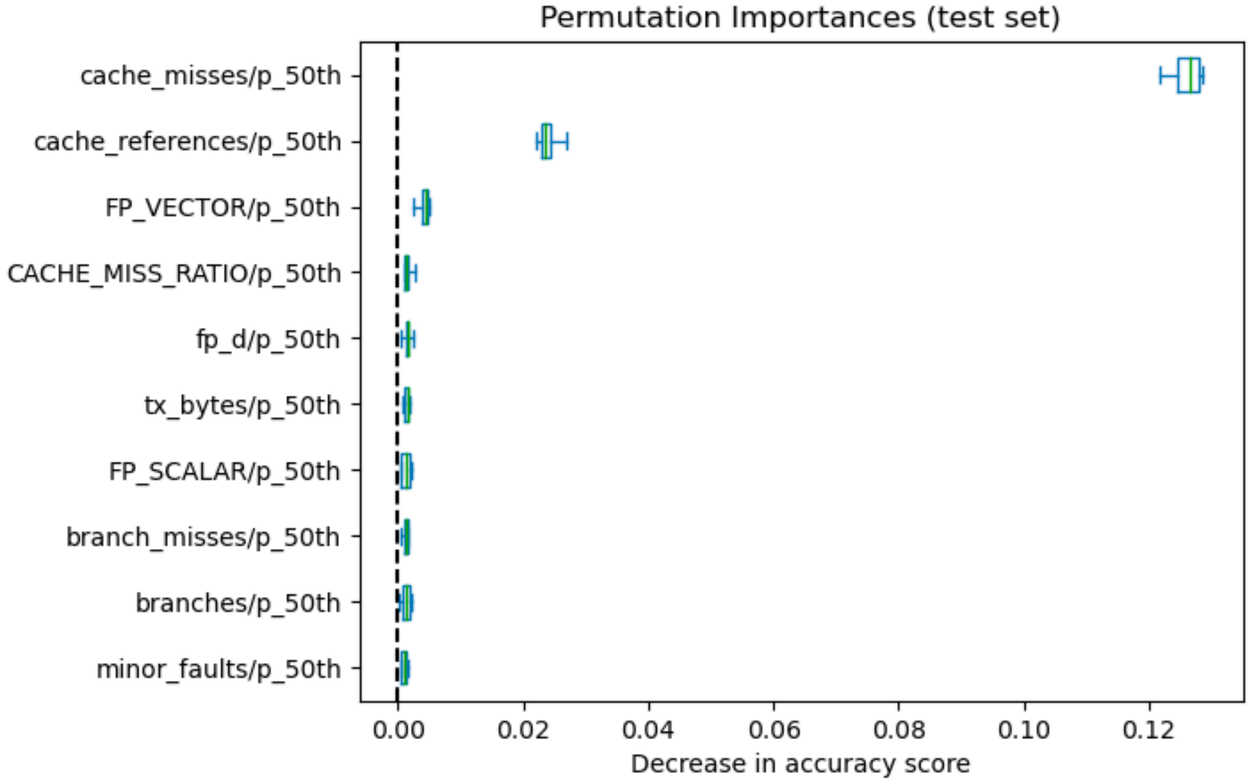


Which Feature is Important

For the Random Forest (CPU)

- Feature Importance and Permutation Importance



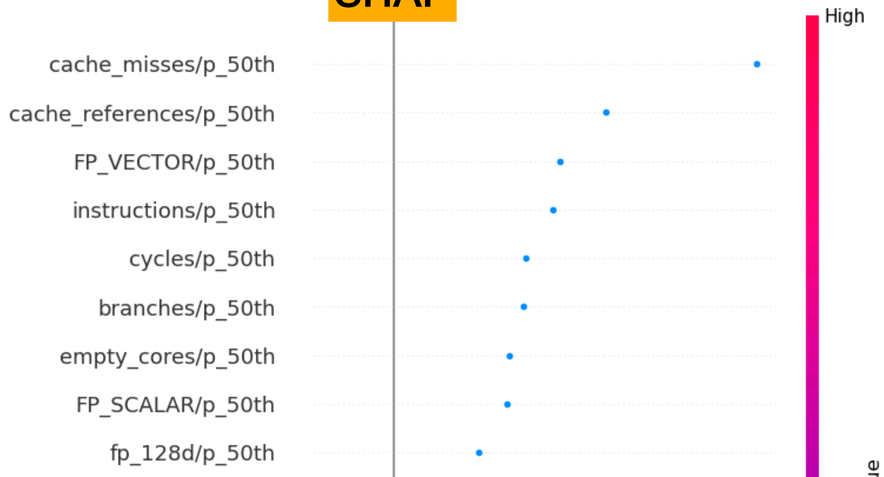


Which Feature is Important

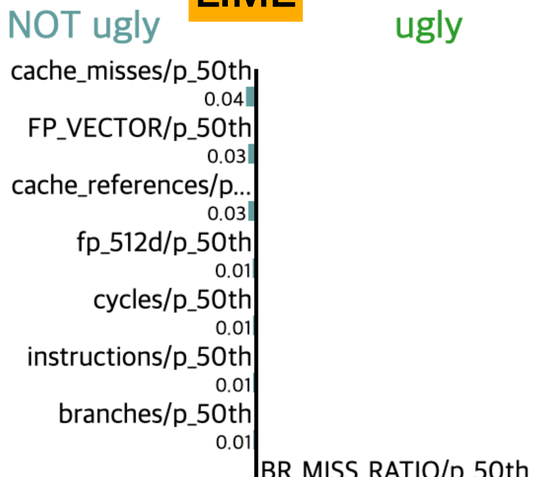
For specific job (CPU)

- Use several methods and compare the result
- **SHAP**: the average expected marginal contribution of one feature
- **LIME**: Local Interpretable Model-Agnostic Explanations. train explainable model that gives similar result to original model in the local
- **Random method** (proposed by Sebastian): put randomized values into each column, iterate several hundred times, and calculate mean reduction in prediction probability

SHAP



LIME

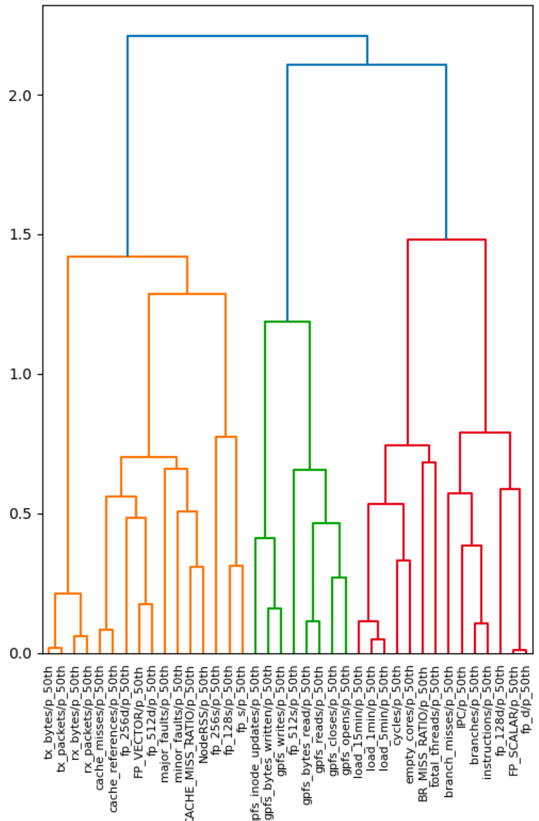


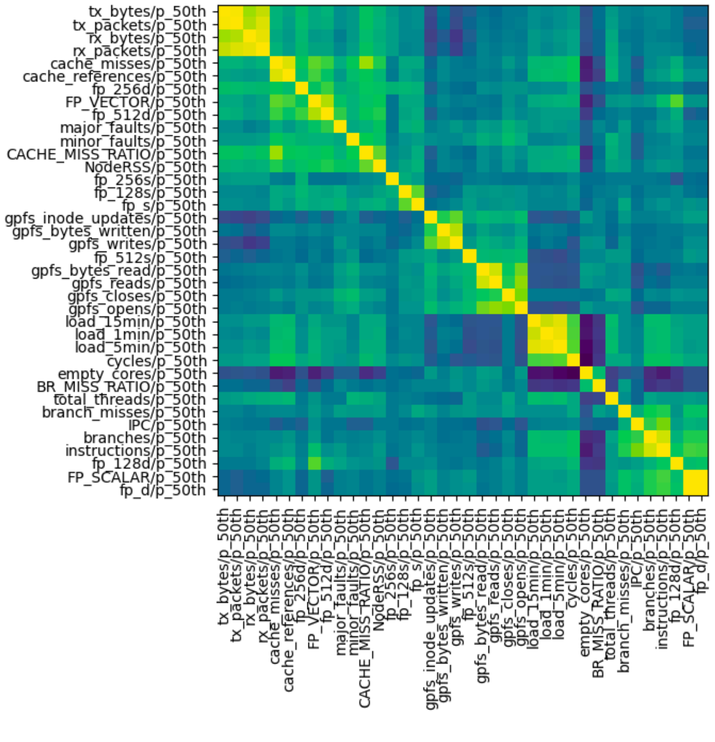
Random Method

cache_misses/p_50th	0.553806
FP_VECTOR/p_50th	0.736231
cache_references/p_50th	0.757374
FP_SCALAR/p_50th	0.815483
fp_256d/p_50th	0.826094
fp_s/p_50th	0.847779
fp_128d/p_50th	0.852184
CACHE_MISS_RATIO/p_50th	0.854234
fp_512d/p_50th	0.861862
gpfs_opens/p_50th	0.873710

Features seem to be correlated

- Features seem to be highly correlated —> Group features with correlation coefficient
  - by performing hierarchical clustering on the Spearman rank-order correlations



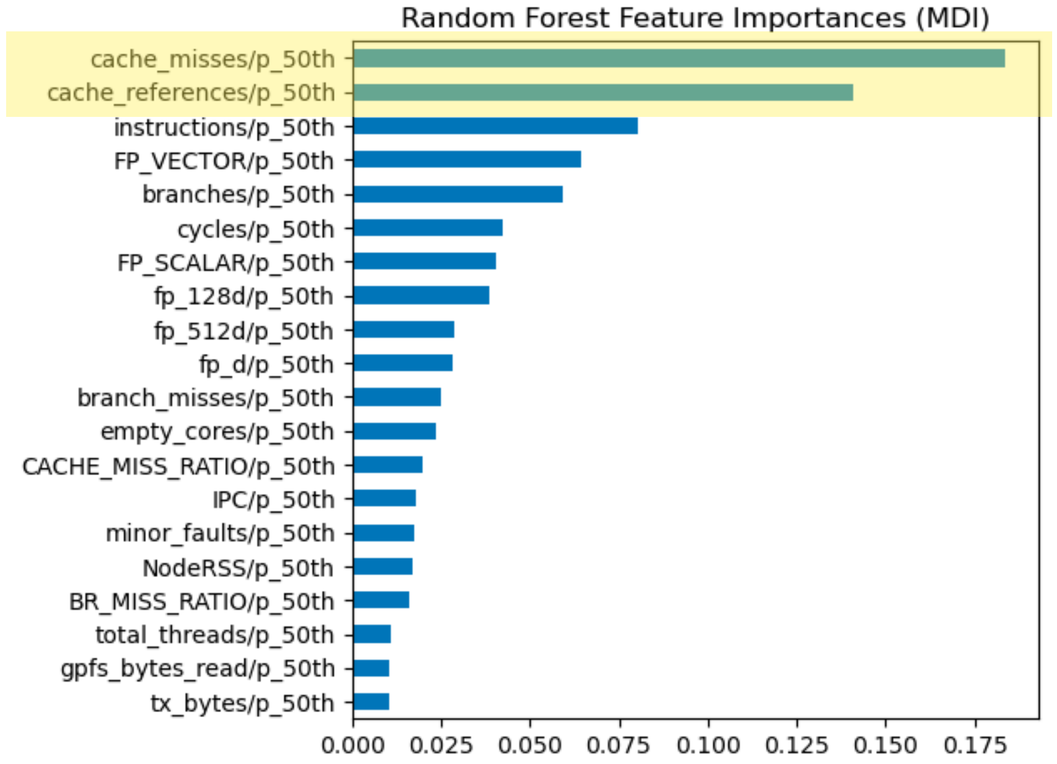


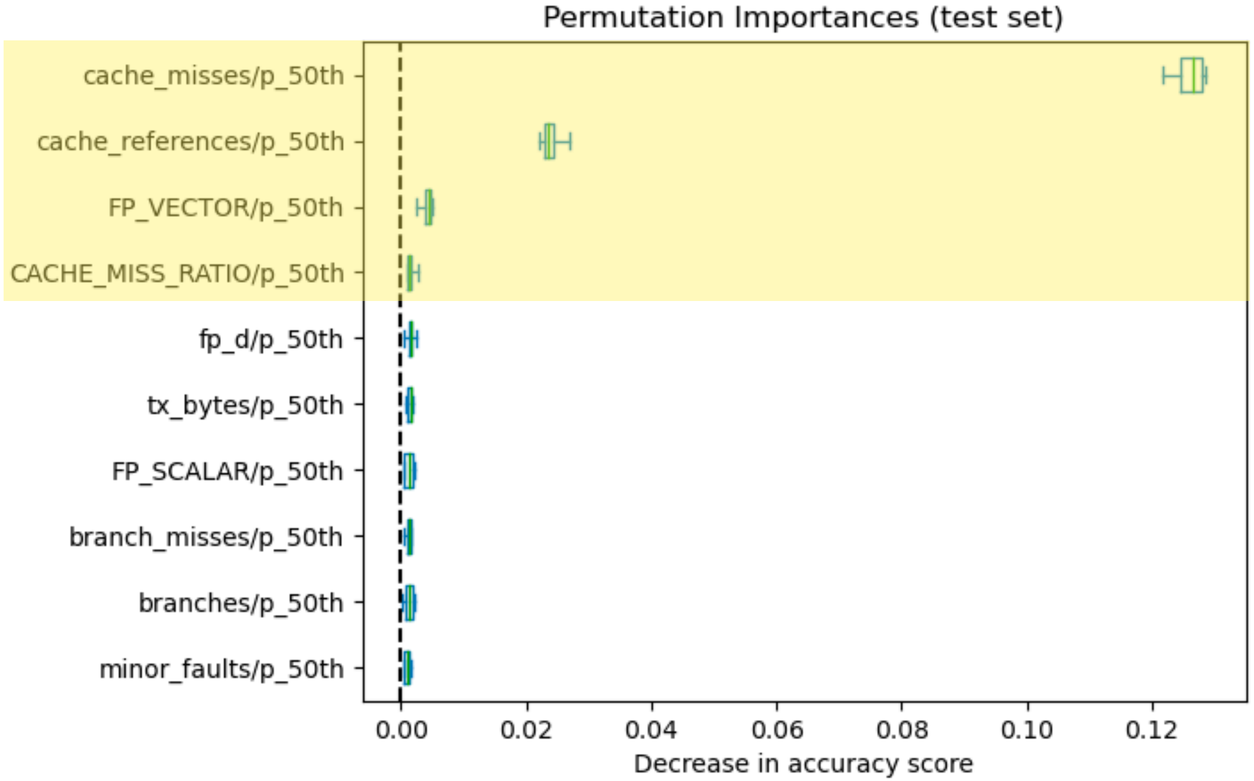

```
['BR_MISS_RATIO/p_50th', 'total_threads/p_50th'], dtype='object'),
['CACHE_MISS_RATIO/p_50th', 'NodeRSS/p_50th', 'major_faults/p_50th',
 'minor_faults/p_50th'],
dtype='object')                                                    ,
['FP_SCALAR/p_50th', 'fp_128d/p_50th', 'fp_d/p_50th'], dtype='object'),
['FP_VECTOR/p_50th', 'cache_misses/p_50th', 'cache_references/p_50th',
 'fp_256d/p_50th', 'fp_512d/p_50th'],
dtype='object')                                                          ,
['IPC/p_50th', 'branch_misses/p_50th', 'branches/p_50th',
 'instructions/p_50th'],
dtype='object')                                                  ,
['cycles/p_50th', 'empty_cores/p_50th', 'load_15min/p_50th',
 'load_1min/p_50th', 'load_5min/p_50th'],
dtype='object')                                                    ,
['fp_128s/p_50th', 'fp_s/p_50th'], dtype='object'),
['fp_256s/p_50th'], dtype='object'),
['fp_512s/p_50th', 'gpfs_bytes_read/p_50th', 'gpfs_closes/p_50th',
 'gpfs_opens/p_50th', 'gpfs_reads/p_50th'],
dtype='object')                                                        ,
['gpfs_bytes_written/p_50th', 'gpfs_inode_updates/p_50th',
 'gpfs_writes/p_50th'],
dtype='object')                                                    ,
['rx_bytes/p_50th', 'rx_packets/p_50th', 'tx_bytes/p_50th',
 'tx_packets/p_50th'],
dtype='object')                                                      ],
```

Which Feature is Important

For the Random Forest

- Feature Importance and Permutation Importance

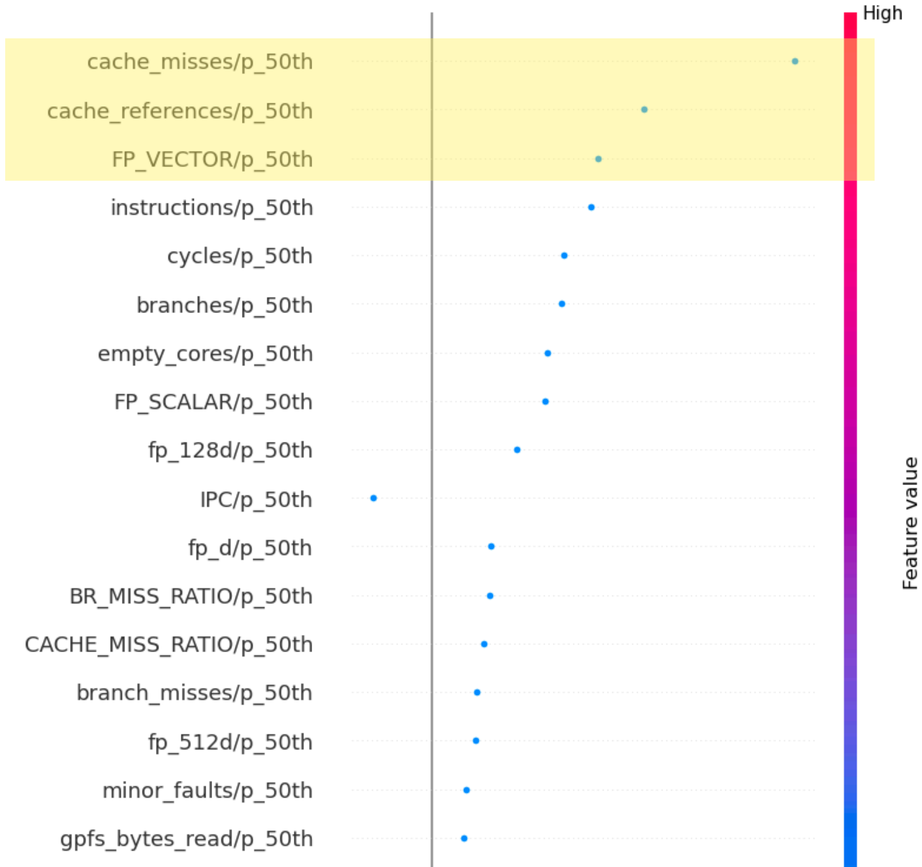




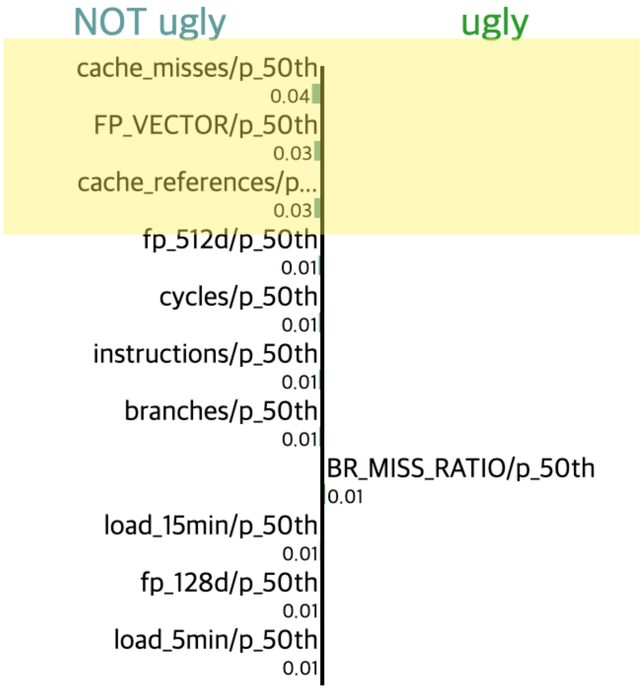
Which Feature is Important

For specific job

SHAP



LIME

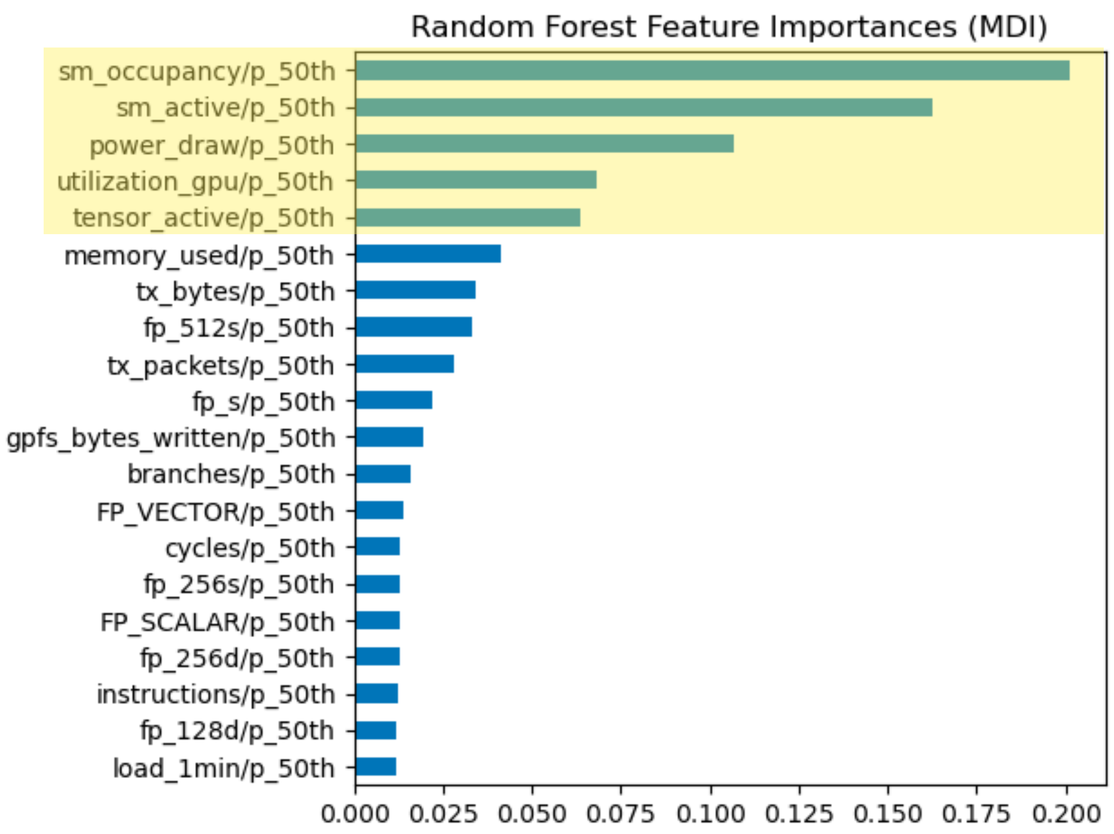


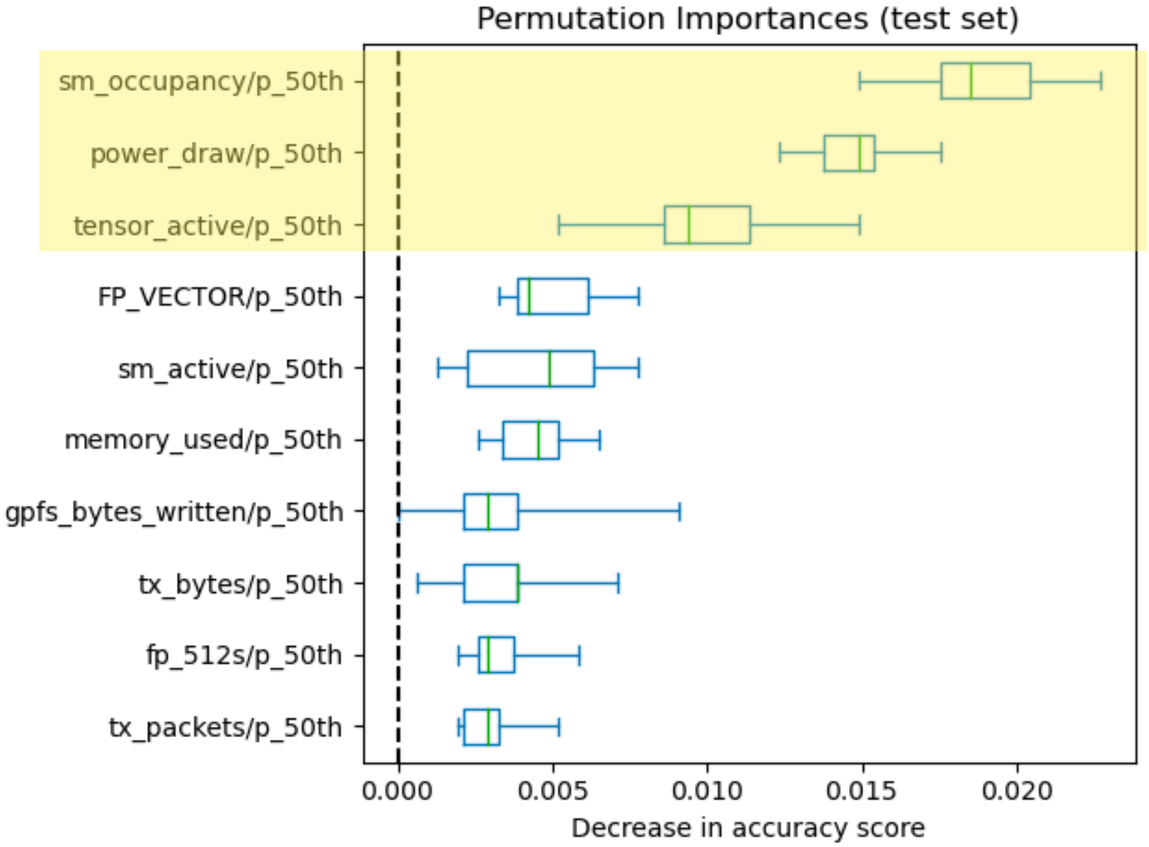
Random Method

cache_misses/p_50th	0.553806
FP_VECTOR/p_50th	0.736231
cache_references/p_50th	0.757374
FP_SCALAR/p_50th	0.815483
fp_256d/p_50th	0.826094
fp_s/p_50th	0.847779
fp_128d/p_50th	0.852184
CACHE_MISS_RATIO/p_50th	0.854234
fp_512d/p_50th	0.861862
gpfs_opens/p_50th	0.873710

GPU Result

For the Random Forest





Which Feature is Important

For specific job

- SHAP result for all jobs

	0	1	2	3	4	5	6
7297039	sm_occupancy/p_50th	sm_active/p_50th	power_draw/p_50th	utilization_gpu/p_50th	tensor_active/p_50th	fp_512s/p_50th	fp_s/p_50th
7353442_11	sm_occupancy/p_50th	sm_active/p_50th	power_draw/p_50th	utilization_gpu/p_50th	tensor_active/p_50th	branches/p_50th	fp_s/p_50th
7353448_5	power_draw/p_50th	tensor_active/p_50th	branches/p_50th	sm_occupancy/p_50th	tx_bytes/p_50th	fp_512s/p_50th	instructions/p_50th
7335476	sm_occupancy/p_50th	sm_active/p_50th	utilization_gpu/p_50th	power_draw/p_50th	tensor_active/p_50th	fp_512s/p_50th	branches/p_50th
7175693	sm_occupancy/p_50th	sm_active/p_50th	utilization_gpu/p_50th	power_draw/p_50th	tensor_active/p_50th	fp_s/p_50th	tx_bytes/p_50th
...	...	...	...	...	...	...	...
7353442_4	sm_occupancy/p_50th	sm_active/p_50th	power_draw/p_50th	tensor_active/p_50th	tx_bytes/p_50th	branches/p_50th	fp_s/p_50th
7274398	sm_occupancy/p_50th	sm_active/p_50th	power_draw/p_50th	utilization_gpu/p_50th	tensor_active/p_50th	tx_bytes/p_50th	fp_512s/p_50th
7351613	sm_occupancy/p_50th	sm_active/p_50th	power_draw/p_50th	utilization_gpu/p_50th	tensor_active/p_50th	fp_512s/p_50th	fp_128d/p_50th
7334894	sm_occupancy/p_50th	sm_active/p_50th	power_draw/p_50th	utilization_gpu/p_50th	tensor_active/p_50th	tx_bytes/p_50th	fp_512s/p_50th
7230186	sm_occupancy/p_50th	sm_active/p_50th	utilization_gpu/p_50th	power_draw/p_50th	tensor_active/p_50th	minor_faults/p_50th	fp_512s/p_50th

Additional testing

- Check if Random Forest can predict (i.e., check if it gives consistent result even though job is not over)
- Train only with two or three columns (cache_miss_ratio or FP_VECTOR) —> test accuracy over 95% (original model: 0.982)
- Other methods: SVM(Support Vector Machine), kNN(k Nearest Neighbors)
  - Similar, but different results: all affected by ‘one specific column’, but that is different
- Check sensitivity of the Random Forest
  - Change number of trees, minimum data number for nodes or to split, etc.
  - Result: Always got similar result —> Model is Robust

Conclusion

- identification of 'good' or 'bad' jobs: by Roofline Model
- identification of the bottleneck for specific jobs and applications: by Random Forest, one feature important, dominant for the performance rating
- fingerprinting of unknown applications, e.g. shedding light at the famous 'a.out' or 'python' mystery: by PCA can differentiate types of executables
- Additionally, neatly modularized all codes I used, so that further studies can be continued

Further Direction

- Check why different methods (SVM, kNN) gives different results. Check prediction quality of those methods
- Look at the data itself thoroughly and see if they are all affected by ‘cache_misses’

What I learned here

- Things related to computer science
  - Concepts of computer architecture and computer performance
  - Multi-processing
  - What I can do with models (We used model not only to just ‘predict’, but also to interpret high dimensional data)
  - How to interpret results from models (feature importances, SHAP, Lime etc.)
  - What Supercomputer looks like!
- My English got better :)
- Meet new people from diverse countries. It was very fun listening to different stories and share experiences!