|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ARE | F1 Score | WMRE | RE | FSE |
| Flow Size Estimation | 🗸 |  |  |  |  |
| Heavy Hitter Detection |  | 🗸 |  |  |  |
| Heavy Change Detection |  | 🗸 |  |  |  |
| Estimation of Flow Size Distribution |  |  | 🗸 |  |  |
| Estimation of Flow Size Entropy |  |  |  | 🗸 |  |
| Estimation of Flow Cardinality |  |  |  | 🗸 |  |
| Estimation of Flow Set |  |  |  |  | 🗸 |

FSE: Flow Set Estimation

近期计划：

1. 将文章发给国外的老师看一下
2. 使用模拟器补充实验
3. 使用mininet补充实验
4. 使用p4交换机补充吞吐率的实验
5. 抽空看一下TurboFlow[EuroSys’18]
6. 写论文
7. The comparison of performance between a large counting table and a hierarchical counting table.

Experiments Plan

|  |  |  |
| --- | --- | --- |
| ID | Description | Status |
| 84414 | Set the memory size to be 1MB. Increase the number of flows from 10K to 250K. Calculate the FSC of flow monitoring, the RE of cardinality and the ARE and F1 score of heavy hitter detection. | Done |
| 84435 | Set the memory size to be 1MB and set the number of flows to be 50K. Replay 10 different trace files. Calculate the ARE of flow size estimation and FSC of flow monitoring. | Done |
| 84442 | Perform this experiment using a software simulator. Set the number of flows as well as the number of cells of the hash table to be 50K. Compare the cache hit ratio of a hashing table with multiple hash functions against hierarchical hashing tables with multiple hash functions. The value of alpha of the hierarchical hash functions should be increased from 0.1 to 1.0. Each hash table will be used for 10 sets of random flows. | Done |
| 84448 | Perform the experiment using a software simulator. Increase the number of flows as well as the number of cells of the hashing table from 20K to 200K. Calculate the cache hit ratio of a normal hashing table and three hierarchical hashing tables. The value of alpha corresponding to the hierarchical hashing tables should be 0.6, 0.7 and 0.8 respectively. | Done |
| 84454 | Compare the performance of HashFlow with a normal hashing table against that of HashFlow with a hierarchical hashing table (with the alpha value of 0.6, 0.7, 0.8 respectively). Set the memory size to be 1MB. Increase the number of flows from 10K to 250K. Calculate the FSC of flow monitoring and F1 score of heavy hitter detection. | Done |
| 84475 | For each of the 10 trace files, extract the first 50K flows and prepare the packets in a list. Run the algorithms (python version) as separate applications in two dedicated CPU cores. The memory size of each algorithm is 1MB. Feed the packets into the algorithms sequentially. Calculate the throughput of the algorithms and the average number of memory accesses. | Done |
| 84477 | Make a hash table for each of the 10 trace files. The number of cells in each hash table is the same as the number of flows in the corresponding trace file. Hash the flows of each trace file to the corresponding hash table. Discard the flow when collision occurs. Calculate the ratio of flows that can be successfully hashed to the hash table.  Use two hash functions for the trace files and hash tables. Use the second hash function to hash the flow to the hash table if collision occurs for the first hash function. Discard the flow only when collision occurs for both hash functions.  Increase the number of hash functions to 3 and 4 and repeat the above process. | Done |
| 84478 | Calculate the flow size distribution of trace file 1. | Done |
| 84479 | HashFlow-d1, HashFlow-d2, HashFlow-d3, HashFlow-d4 are the versions of HashFlow whose main table consisting of 1, 2, 3 and 4 sub-tables respectively (the value of alpha is 0.6). We replay the 10 trace files for each of the algorithms and calculate the FSC of flow monitoring, F1 score of heavy hitter detection, ARE of flow size estimation and RE of cardinality estimation. | Done |
| 84480 | For each of the 10 trace files, extract the first 50K flows and prepare the packets in a list. Run the algorithms as separate applications in two dedicated CPU cores. The memory size of each algorithm is 1MB. Feed the packets into the HashFlow-d1, HashFlow-d2, HashFlow-d3, HashFlow-d4 sequentially. Calculate the throughput of the algorithms and the average number of memory accesses. | Done |
| 84481 | Set the number of cells and the number of flows to be 100000. Increase the number of hashes from 1 to 10. Calculate the utilization ratio of the hash table. |  |
| 84482 | Create a count-min sketch with 4 rows and 1000 columns. Feed a number of flows into the sketch. The number of flows feed into the sketch is increased from 1000 to 10000, with the step of 1000. Query the sketch for the sizes of the flows and calculate the average relative error of the flows. |  |
| 84483 | Create a hierarchical of 3 layers. The size of the hash table is 100K. There is a hash function associated with each layer. We increase the value of parameter alpha from 0.1 to 1.0. We feed 100K random strings into the hash table and calculate the utilization of the hash table. We will also organize the hash table as a single large hash table. Then we feed the same 100K random strings and calculate the utilization of the hash table. |  |
| 84484 | Choose a trace file from the campus network trace, CAIDA trace and China Telecom respectively. For each file extract the first 50K flows. Run the algorithms (P4 version) on bmv2 as separate applications in three dedicated CPU cores. The memory size of each algorithm is 1MB. Feed the packets into the algorithms sequentially. Calculate the throughput of the algorithms and the average number of memory accesses. There will be 10 rounds of the experiment for each trace file. | Done |
| 84485 | Memory size: 1 MB  Number of flows: increase from 10K to 250K in the step of 10K.  Traces: Tsinghua trace, Telecom trace, HGC trace, CAIDA trace  Applications: cardinality estimation, flow monitoring | Done |
| 84486 | Memory size: 1 MB  Number of flows: [50K, 100K, 150K, 200K, 250K]  Traces: Tsinghua trace, Telecom trace, HGC trace, CAIDA trace  Threshold for heavy hitters: choose 10 different points according to the characteristics of the traces  Applications: heavy hitter detection  Metrics: F1 Score and ARE | Done |
| 84487 | Memory size: 2.5 MB  Number of Flows: 10K to 250K with the step of 10K  Traces: Tsinghua trace, Telecom trace, HGC trace, CAIDA trace  Application: flow size estimation  Metrics: ARE |  |

During flooding attacks router memory and network bandwidth consumed by flow records can increase beyond what is available

Give a figure to illustrate the processing for a packet

Give a upper-bound of amortized number of memory accesses for storing a packet

Plot the flow size distribution of the trace file

Counter Braids is worth reading.

Read “Space-code bloom filter for efficient per-flow traffic measurement” and write it into the related work.

Unlike admit-all-count-some, we do not automatically admit new flows into our data structure and the vast majority of mouse flows will be by-passed.

“Finding frequent items in data streams.” Is worth of reading.

The CM sketch has an over-estimation problem, which will become severe in a tight memory space where the number of counters is far smaller than the number of flows, resulting in aggressive sharing.

The flows under measurement can be per-source flows, per-destination flows, per-source/destination flows, TCP flows, http flows, or any user-defined logical flows.

Moreover, the limited on-chip SRAM is shared by different functions, such as routing, scheduling, traffic measurement, and security.

It will be good if we can consult the vendor about the memory size and time budget that can be allocated to the measurement algorithms.

A flow tag can be an IP address, a source and destination IP address pair, the combination of IP addresses and port numbers, or other attributes that can identify a flow.

Paper Layout:

|  |  |
| --- | --- |
| Section | Contents |
| Introduction | Background introduction.  Introduction to NetFlow  Problem description.  Challenges.  My solution and basic ideas  Experimental results overview  Paper layout. |
| Background and Motivation | Introduction to three types of algorithms: sampling based algorithms, hash table based algorithms and sketch based algorithms  Introduction to HashPipe, ElasticSketch and FlowRadar  The drawbacks of the algorithms  Motivation  Why to discard information regularly  Why to cache the elephant flows preferentially  The computation/memory utilization tradeoff |
| Evaluation | The performance of HashFlow for various depths of MT.  The performance of HashFlow for various values of alpha  The ARE and F1 score of heavy hitter detection, FSC of flow monitoring, and RE of cardinality estimation of HashFlow, HashPipe, ElasticSketch and FlowRadar when the number of flows increases from 10K to 250K.  The FSC of flow monitoring and ARE of flow size estimation for the 10 trace files.  The throughput of HashFlow, HashPipe, ElasticSketch and FlowRadar. |