Balancing Fairness and Efficiency for Cache Sharing in Semi-external Memory System

Shanjiang Tang[1], Qifei Chai[1], Ce Yu[1], Yusen Li[2], Chao Sun[1]

[1]College of Intelligence and Computing, Tianjin University

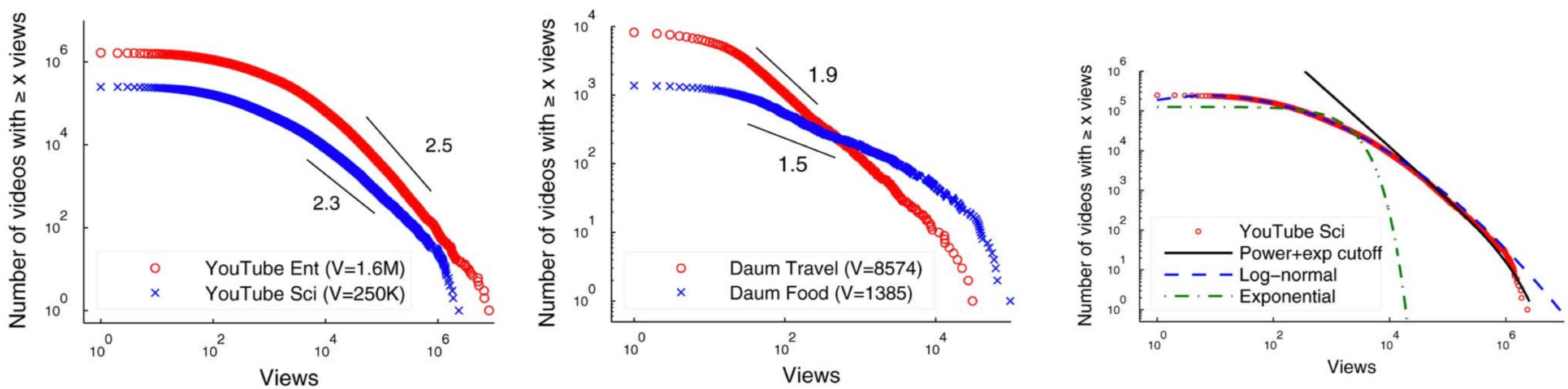
[2]School of Computer Science, Nankai University

Outline

- Motivation
- Elastic Semi-External Memory Allocation
- Evaluation
- Conclusion and Future Work

Data Caching is Important

- There are varied accesses frequencies for applications data.
  - Many real applications follow *power-law* distribution for their data accesses.
  - Put hot data in cache can speedup the performance.



(a) YouTube empirical plot (b) Daum empirical plot (c) Curve fitting for YouTube

Fig. 4. Video popularity distributions of YouTube and Daum videos follow a power-law distribution in the waist with exponents between 1.5 and 2.5. YouTube Sci and Daum Food exhibit decays in the tail of their distributions, which represents the most frequently viewed content.

Cha et al. Analyzing the Video Popularity Characteristics of Large-Scale User Generated Content Systems, TON'09.

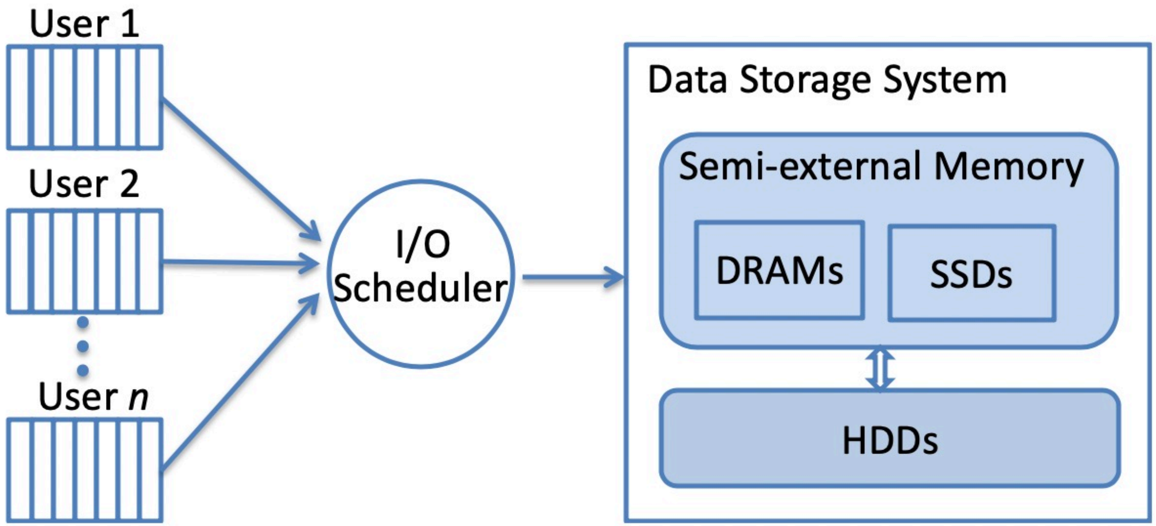
Cache Sharing is a Trend

- Cache sharing can improve the cache efficiency.
  - Allow overload users to use the idle cache resources from underloaded users for **maximum cache utilization**.
  - Keep only **one copy** of shared data for multiple users.
  - Enable **global efficiency optimization** across multiple users.
  - Supported by many existing cache systems for caching data in **DRAM** for fast data access.

The capacity of DRAM is **limited** for big data caching!

Semi-External Memory (SEM) Cache Model

- Overcome the capacity limitation of DRAMs by adding SSDs.
  - Data can be cached either in DRAMs or SSDs.
  - The latency of DRAMs is much smaller than SSDs.
  - *Cache Hit*: an access to DRAMs or SSDs
  - *Cache Miss*: an access to HDDs.

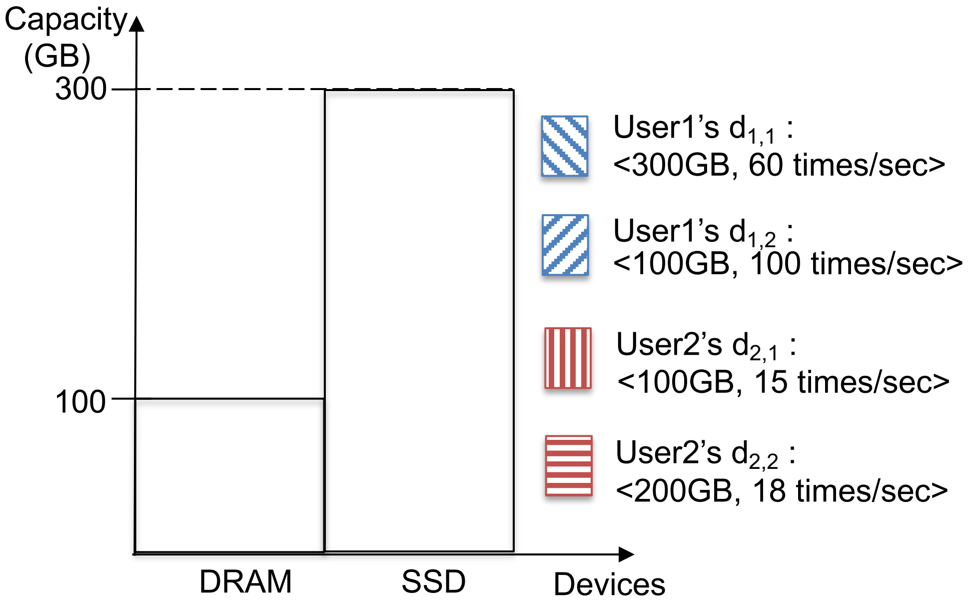


Cache Resource Allocation

- Integrate DRAMs and SSDs of SEM with the awareness of their different data access latencies.
  - If latency ratio of DRAM to SSDs is 1:6, then 1GB DRAM can trade for 6GB SSD.
  - Users care about the total allocated cache resources of all storage devices in SEM, rather than separately.
- Different allocation policies can have different allocation results on **Fairness** and **Efficiency**.
  - Global Sharing Policy (e.g., LFU)
  - Separate Max-min Fairness Policy
  - Global Max-min Fairness Policy

Motivating Example

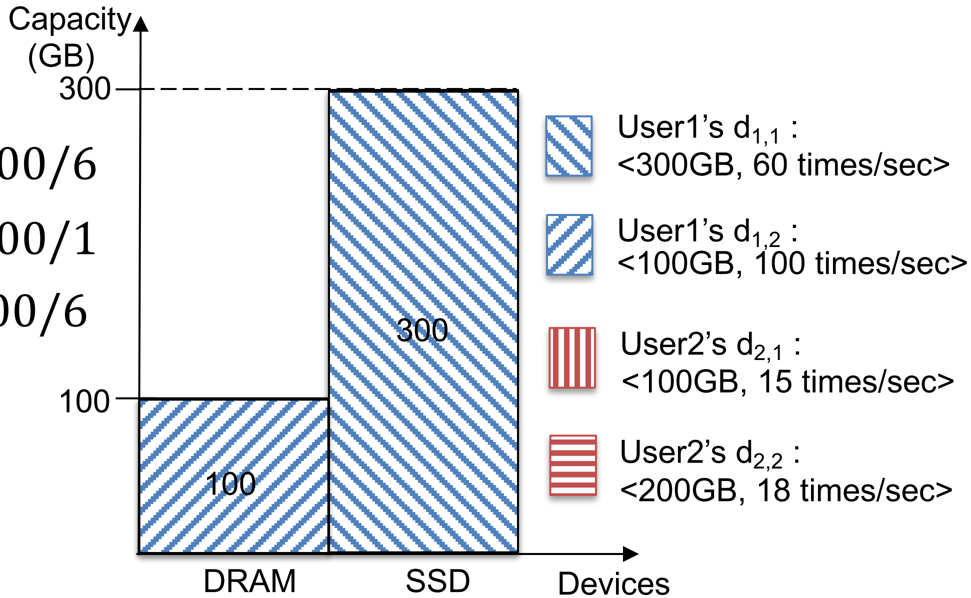
- Consider a SEM consisting of 100 GB DRAM and 300 GB SSD, where the latency ratio of DRAM to SSD is 1/6. It is shared by two users 1 and 2 equally. User 1 contains two data $d_{1,1}$ (size: 300 GB, access frequency: 60 times/sec) and $d_{1,2}$ (size: 100 GB, access frequency: 100 times/sec). User 2 has two data $d_{2,1}$ (size: 100 GB, access frequency: 15 times/sec) and $d_{2,2}$ (size: 200 GB, access frequency: 18 times/sec).



Global Sharing Policy (e.g., LFU)

- Consider a SEM consisting of 100 GB DRAM and 300 GB SSD, where the latency ratio of DRAM to SSD is 1/6. It is shared by two users 1 and 2 equally. User 1 contains two data $d_{1,1}$ (size: 300 GB, access frequency: 60 times/sec) and $d_{1,2}$ (size: 100 GB, access frequency: 100 times/sec). User 2 has two data $d_{2,1}$ (size: 100 GB, access frequency: 15 times/sec) and $d_{2,2}$ (size: 200 GB, access frequency: 18 times/sec).

- Allocation results
  - User1's Allocation:150=100/1+300/6
  - User1's Efficiency: 13000=100*100/1 +60*300/6
  - User2's Allocation: 0
  - User2's Efficiency: 0
  - Total efficiency: 13000

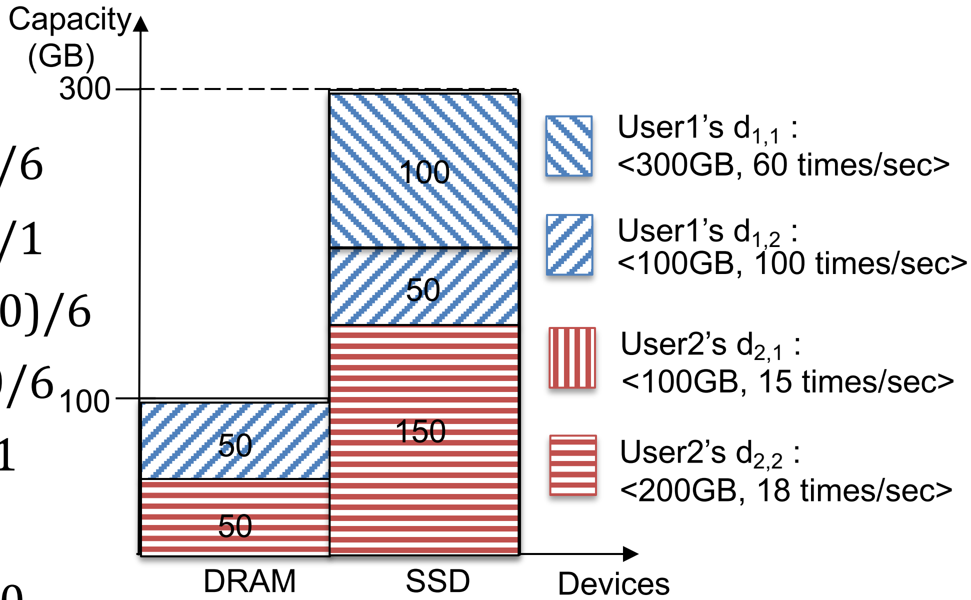


Unfairness Degree: |150/75 – 0/75| = 2, SEM efficiency:13000 .

Separate Max-min Fairness Policy

- Consider a SEM consisting of 100 GB DRAM and 300 GB SSD, where the latency ratio of DRAM to SSD is 1/6. It is shared by two users 1 and 2 equally. User 1 contains two data $d_{1,1}$ (size: 300 GB, access frequency: 60 times/sec) and $d_{1,2}$ (size: 100 GB, access frequency: 100 times/sec). User 2 has two data $d_{2,1}$ (size: 100 GB, access frequency: 15 times/sec) and $d_{2,2}$ (size: 200 GB, access frequency: 18 times/sec).

- Allocation results
  - User1's Allocation:75=50/1+150/6
  - User1's Efficiency: 6833=50*100/1 + (50*100+100*60)/6
  - User2's Allocation: 75=50/1+150/6
  - User2's Efficiency: 1350=50*18/1 +150*18/6
  - Total efficiency: 8183=6833+1350

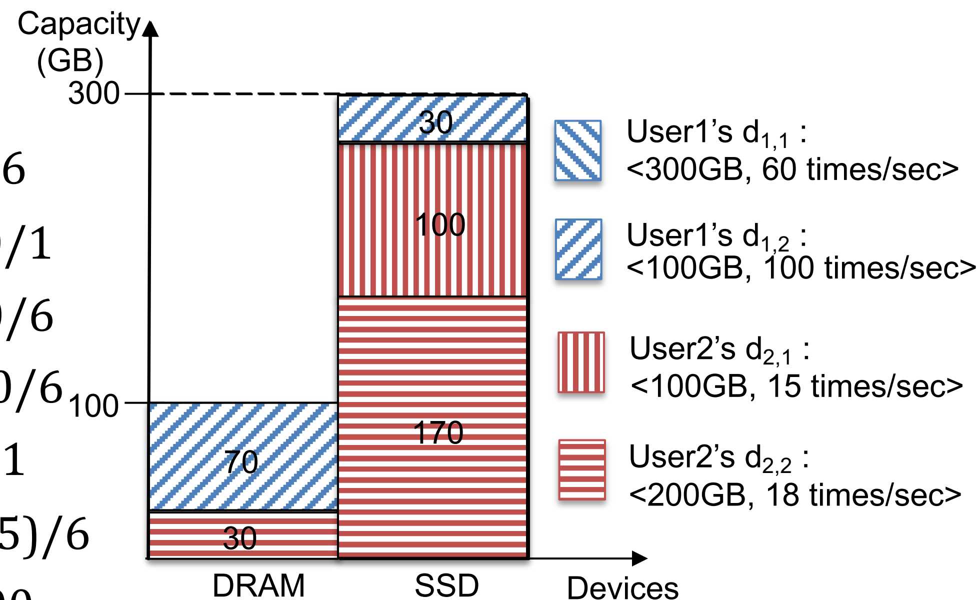


Unfairness Degree: |75/75 – 75/75| = 0, SEM efficiency:8183.

Global Max-min Fairness Policy

- Consider a SEM consisting of 100 GB DRAM and 300 GB SSD, where the latency ratio of DRAM to SSD is 1/6. It is shared by two users 1 and 2 equally. User 1 contains two data $d_{1,1}$ (size: 300 GB, access frequency: 60 times/sec) and $d_{1,2}$ (size: 100 GB, access frequency: 100 times/sec). User 2 has two data $d_{2,1}$ (size: 100 GB, access frequency: 15 times/sec) and $d_{2,2}$ (size: 200 GB, access frequency: 18 times/sec).

- Allocation results
  - User1's Allocation:75=70/1+30/6
  - User1's Efficiency: 7500=70*100/1 + 30*100/6
  - User2's Allocation: 75=30/1+270/6
  - User2's Efficiency: 1300=30*18/1 +(170*18+100*15)/6
  - Total efficiency: 8800=7500+1300



Unfairness Degree: |75/75 – 75/75| = 0, SEM efficiency:8800.

Fairness VS Efficiency

- Tend to be a tradeoff between fairness and efficiency.
  - Pursuing 100% fairness often results in ***poor*** efficiency, and vice versa.
  - Needs an allocation policy that can balance the two metrics flexibly as users want.

Outline

- Motivation
- Elastic Semi-External Memory Allocation
- Evaluation
- Conclusion and Future Work

Elastic Semi-External Memory Allocation

- **Basic Ideas:** trade fairness for increasing cache allocation efficiency with some degree of unfairness.
  - Two terms: strict fairness and relaxed fairness.
    - Strict fairness: 100% fairness between any two users.
    - Relaxed fairness: |allocationOfUser1 – allocationOfUser2| $\leq \theta$
  - Maximize system efficiency while keep the **relaxed** fairness.
- **ElasticSEM:** A combination of two allocations.
  - Fairness-stage allocation. (relaxed fairness guarantee)
  - Efficiency-stage allocation. (efficiency maximization)

Elastic Semi-External Memory Allocation

- Define a Knob σ to balance the fairness and efficiency allocation.
  - Divide users into two sets: **fairnessGuaranteeUserset** and **fairnessNotGuaranteeUserset**
  - A user whose fairness is not satisfied always has the choice to evict and cache data.
  - Priority can be frequency, last access time.

Algorithm 1 Elastic Semi-external Memory Allocation (ElasticSEM).

```
function ElasticSEM(u, d)
    FairnessNOTGuaranteedUserSet = {i ∈ [1, n] | H_i < s_i · σ}.
    FairnessGuaranteedUserSet = {i ∈ [1, n] | H_i ≥ s_i · σ}.
    while DRAM.availableSize + SSD.availableSize < d.size do
        Choose User u' from FairnessGuaranteedUserSet containing a cached data d' of the
lowest priority in SEM.
        if u ∈ FairnessGuaranteedUserSet AND d.priority ≤ d'.priority then
            return CACHE_ABORT.
        else if u = u' and d'.priority > d.priority then
            return CACHE_ABORT.
        else if u ∈ FairnessNOTGuaranteedUserSet OR d.priority > d'.priority then
            if d'.location = DRAM then
                DRAM.availableSize+= d'.size,   d_i^DRAM -= d'.
            else if d'.location = SSD then
                SSD.availableSize+= d'.size,   d_i^SSD -= d'.
    CacheAllocation(u, d).                                   ▷ Cache data d for user u.
```

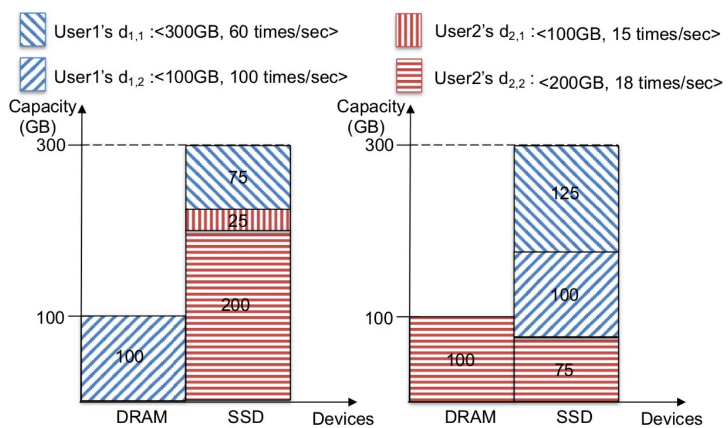
Algorithm 2 Cache allocation function.

```
function CacheAllocation(u, d)
    A = DRAM.availableSize,   D_i = d_i^DRAM.
    A' = SSD.availableSize,   D'_i = d_i^SSD.
    if A ≥ d.size then                                ▷ Cache data d in DRAM of SEM.
        A− = d.size,   D_i+ = d.size.
    else if A < d.size then                 ▷ Cache data d in both DRAM and SSD of SEM.
        Split d into two parts d = {d_1, d_2} satisfying that d_1.size = d.size − A,   d_2 =
d − d_1.
        A− = d_1.size,   A'− = d_2.size,   D_i+ = d_1,   D'_i+ = d_2.
```

Detailed Description is given in the paper.

Cheating Problem for ElasticSEM

- Consider a SEM consisting of 100 GB DRAM and 300 GB SSD, where the latency ratio of DRAM to SSD is 1/6. It is shared by two users 1 and 2 equally. User 1 contains two data $d_{1,1}$ (size: 300 GB, access frequency: 60 times/sec) and $d_{1,2}$ (size: 100 GB, access frequency: 100 times/sec). User 2 has two data $d_{2,1}$ (size: 100 GB, access frequency: 15 times/sec) and $d_{2,2}$ (size: 200 GB, access frequency: 18 times/sec).



(a) ElasticSEM with no cheating (b) ElasticSEM with cheating

Figure 3: ElasticSEM allocation for Example 1 with and without cheating, where the knob $\sigma = 0.5$. In (b), user 2 makes spurious access to $d_{2,2}$ such that its access frequency exceeds $d_{1,2}$, which makes it obtain more resources in Figure 3 (b)(e.g., 100/1+75/6=112.5) than that in Figure 3 (a) (e.g., 25/6+200/6=37.5).

ElasticSEM with Cheating Detection and Punishment Mechanism

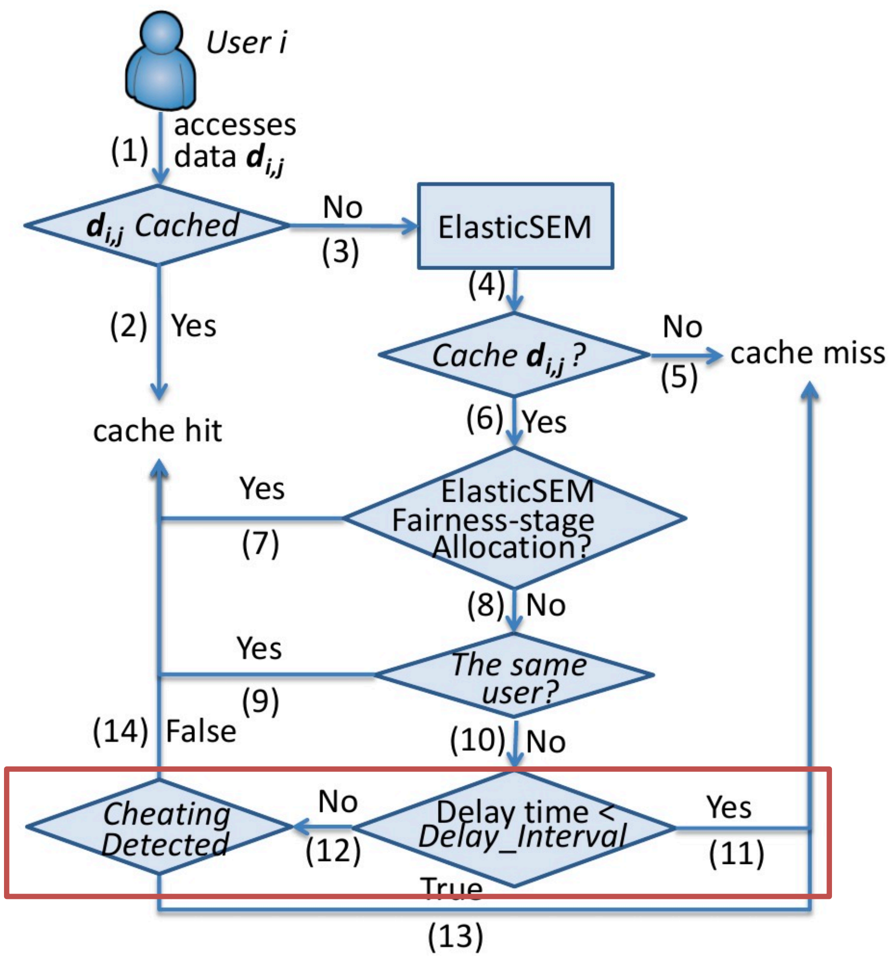


Figure 4: ElasticSEM policy with cheating detection and punishment mechanism.

Detailed Description is given in the paper.

Outline

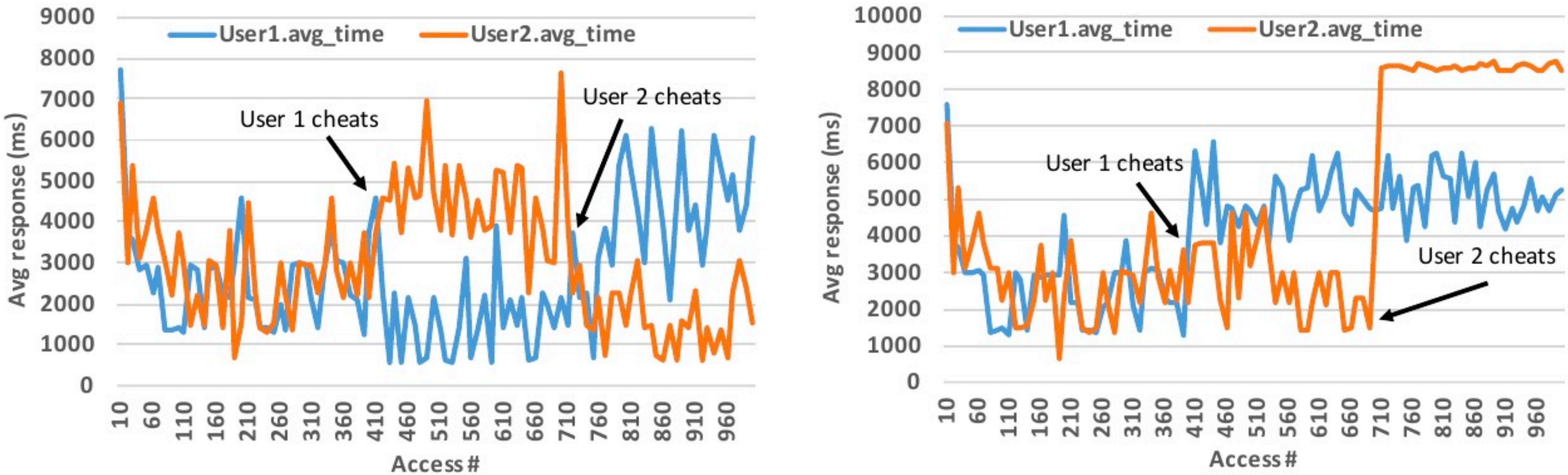
- Motivation
- Elastic Semi-External Memory Allocation
- Evaluation
- Conclusion and Future Work

Evaluation

- Alluxio Cluster
  - 11 nodes, each with 8 CPU cores and 16GB memory.
  - We configure 4GB memory as DRAM cache and use 8GB memory to emulate SSD cache.
- Macro-Benchmarks
  - Three different workloads including synthetic Facebook workload, Purdue workload, TPC-H workload.
- Micro-Benchmarks
  - Two users each with 40 files and equally share the SEM cache resources.

Detailed setups are in the paper.

Cheating and Punishment



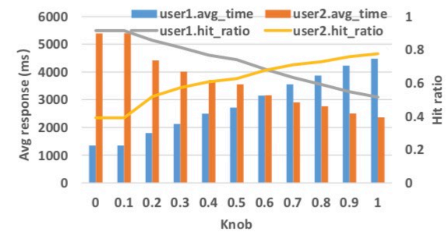
(a) Global resource sharing allocation with LFU.

(b) ElasticSEM allocation with Knob $\rho = 0$.

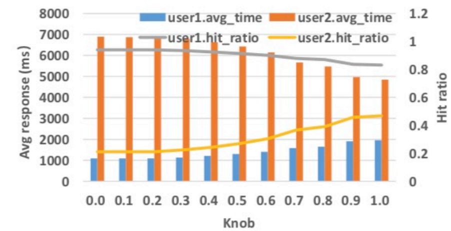
Figure 5: The average response time measured for two users under different allocation policies. User 1 starts cheating at the 400^{th} acess. User 2 started cheating at the 700^{th} access.

Fairness and Efficiency under Different knobs

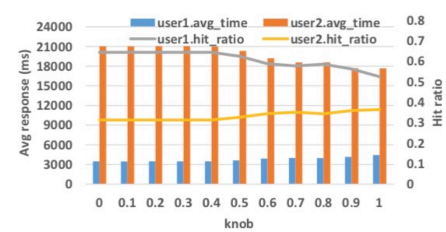
- The system efficiency for User 1 and User 2 under different knobs configurations. The cache volume of SEM system is set to 10GB for DRAM and 30GB for SSD, respectively. We particularly show that the sensitivity of knob configuration on the tradeoff between fairness and efficiency is related to the **cached data distribution** and **their sizes.**



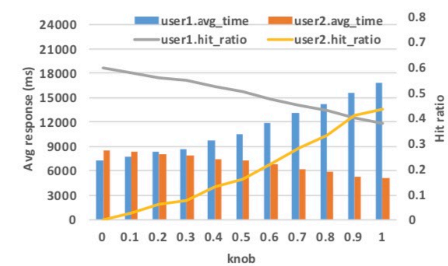
(a) User 1 has 40 files of 1 GB each. It has 5000 data accesses in total and the data access complies with uniform distribution. User 2 has 40 files of 1 GB each. It has 1000 data accesses in total and the data access complies with Zipf distribution.



(b) User 1 has 40 files of 1 GB each. It has 5000 data accesses in total and the data access complies with Zipf distribution. User 2 has 40 files of 1 GB each. It has 1000 data accesses in total and the data access complies with uniform distribution.



(c) User 1 has 40 files of 1 GB each. It has 5000 data accesses in total and the data access complies with uniform distribution. User 2 has 40 files of different sizes. It has 1000 data accesses in total and the data access complies with Zipf distribution, where we assume that its hot data are of large data blocks.



(d) User 1 has 40 files of different Sizes. It has 5000 data accesses in total and the data access complies with Zipf distribution, where we assume that its hot data are of large data blocks. User 2 has 40 files of 1 GB each. It has 1000 data accesses in total and the data access complies with uniform distribution.

Performance Comparison

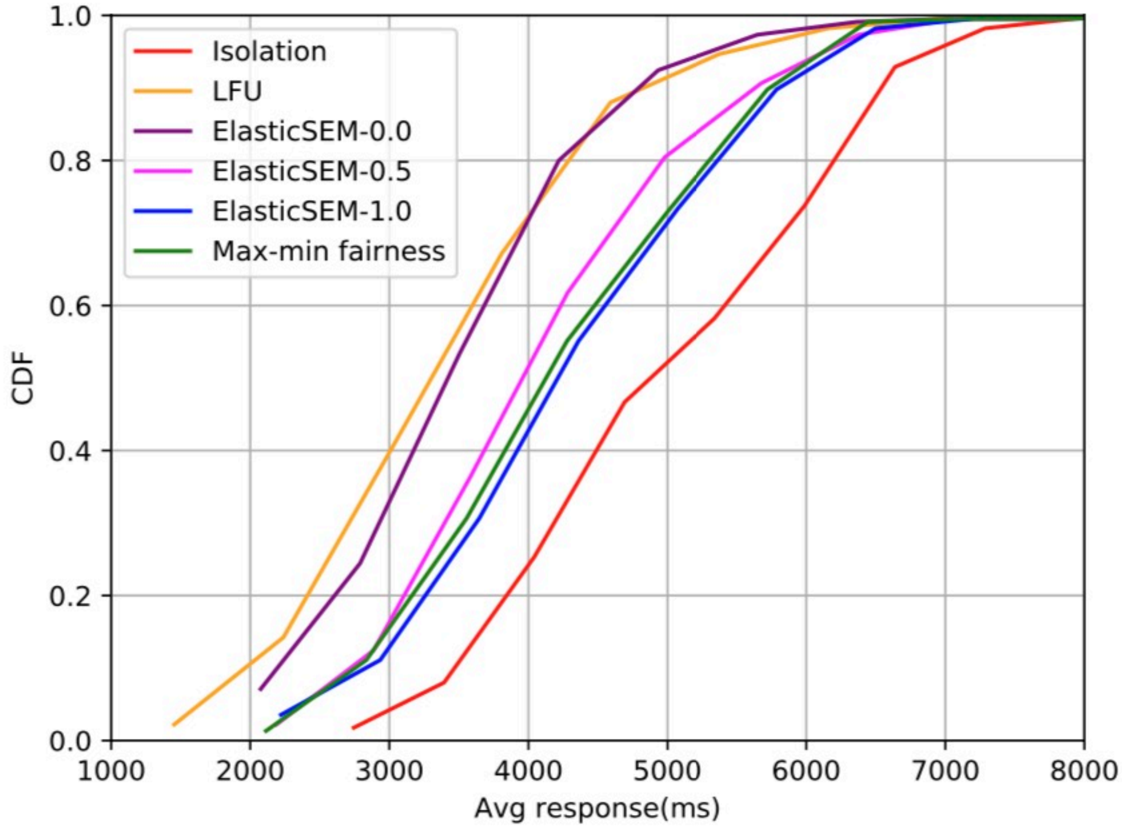


Figure 7: The CDF of average response time for various cache allocation policies.

Outline

- Motivation
- Elastic Multi-Resource Fairness
- Evaluation
- Conclusion and Future Work

Conclusions

- There is a tradeoff between fairness and efficiency for resource allocation in SEM cache system.
- We argue that it should integrate DRAMs and SSDs of SEM as a whole when considering fairness /efficiency optimization in resource allocation.
- We propose a knob-based fairness-efficiency cache allocation policy called ElasticSEM for SEM.
- We experimentally show that ElasticSEM can allow users to balance the tradeoff between fairness and efficiency while addressing the cheating problem.

Thanks!

Question?