Dynamic equation-based memory allocation for stream processing engines

Rengan Dou, Richard T.B. Ma, Y.C. Tay

National University of Singapore

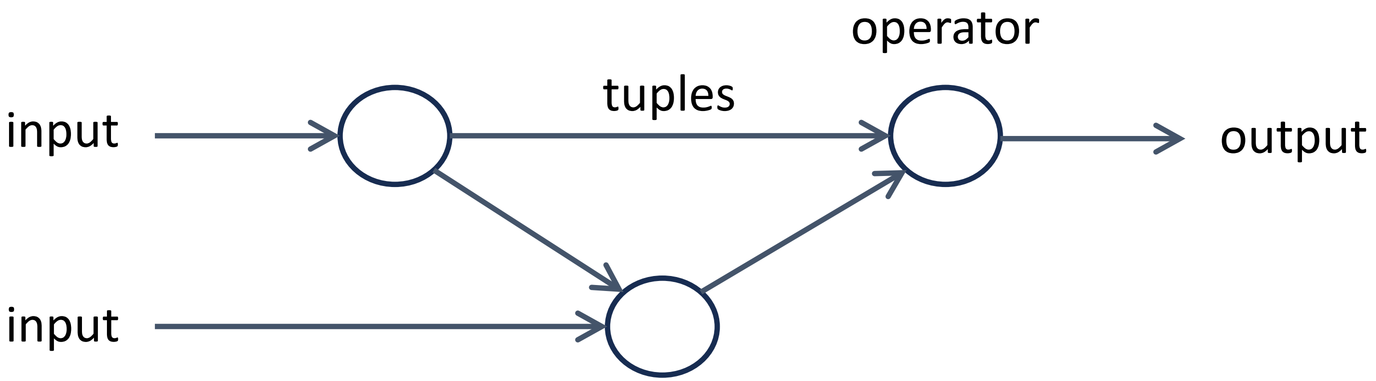
context:

streaming applications (e.g. surgery, autonomous driving, high-frequency trading)

Apache Flink Flink

execution model:

directed acyclic graph (DAG)

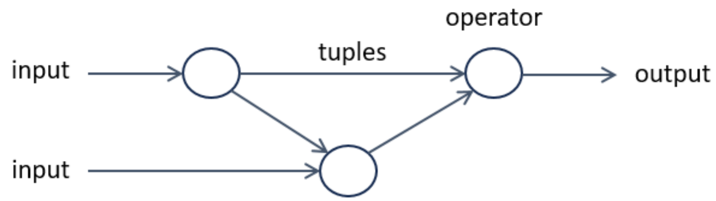


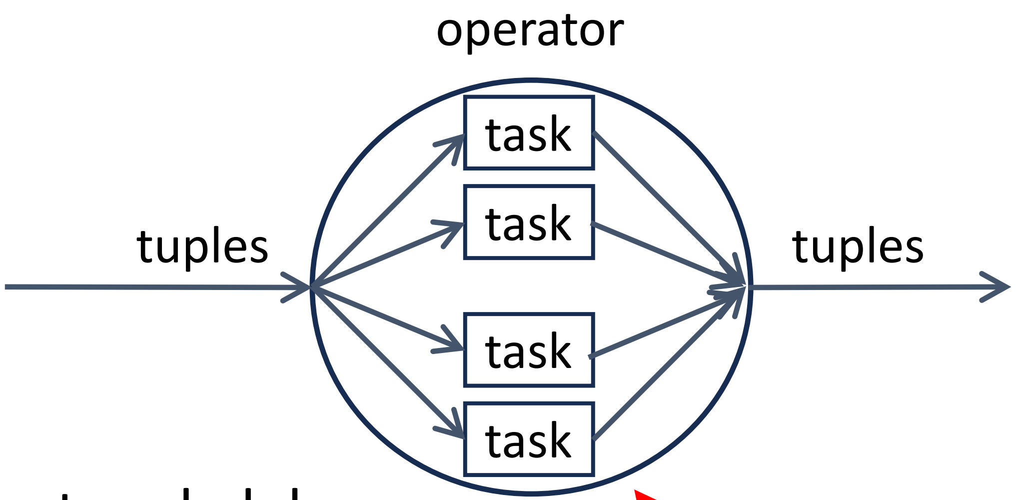
fluctuates over time

latency-sensitive

skewed data distribution

operator:





no operator-to-operator network delay

operator runs parallel tasks (≡ threads)

tuple stream split (unevenly) among tasks

task:

each task has a channel queue

task: stateless/stateful

state (e.g. intermediate results)

saved to disk (persistence)

cached in memory

given: memory size for operator

decide: cache allocation to tasks

cache misses

operator latency

input-output latency

question:

how to allocate operator memory to its tasks?

objective:

$$\underset{\text{path } P \text{ in DAG}}{\textit{minimize}} \quad \sum_{\text{operator } O \text{ in } P} \quad \underset{\text{task } T \text{ in } O}{\textit{maximize}} \quad \textit{latency}(T)$$

solution:

analytic model

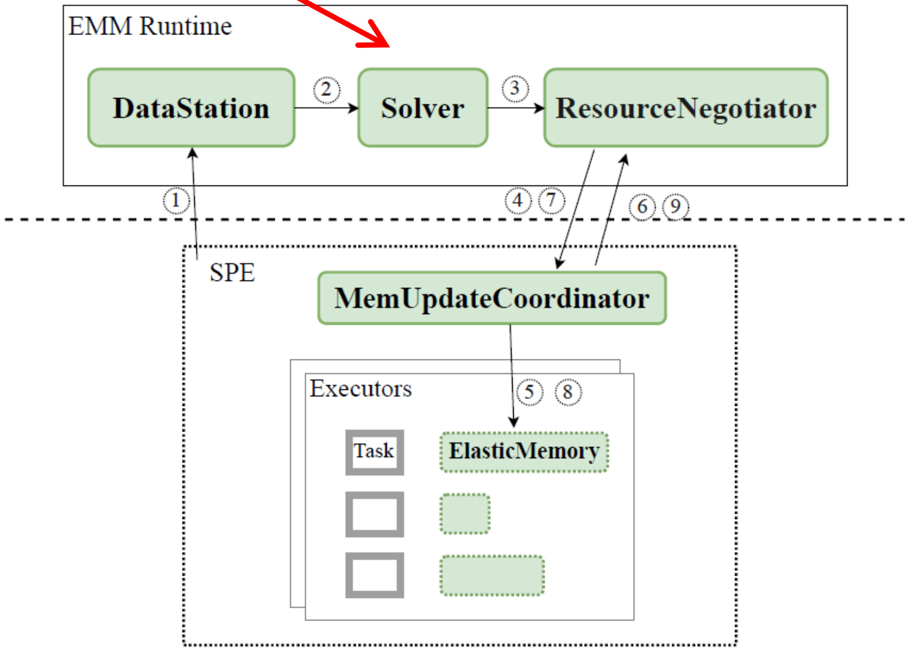
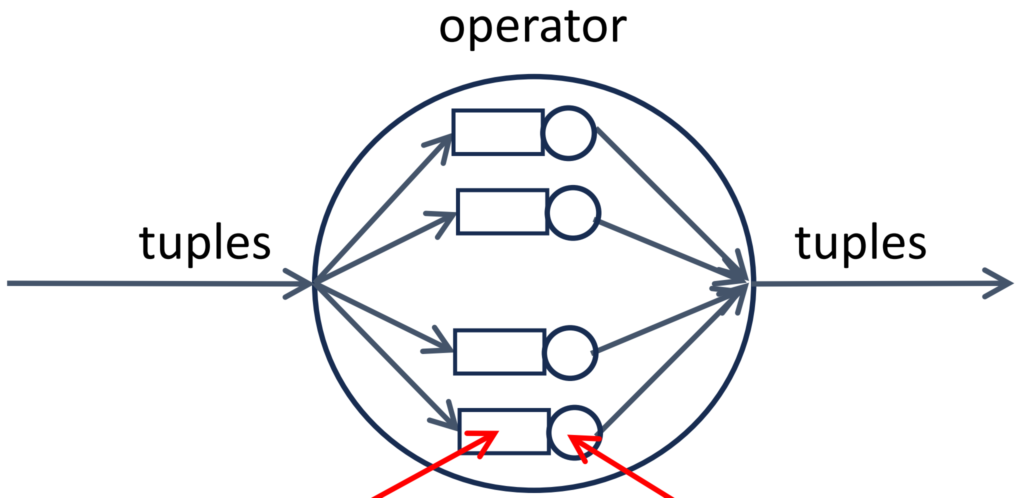


Fig. 4: EMM Framework

Rengan Dou, Richard T. B. Ma: *Latency-Oriented Elastic Memory Management at Task-Granularity for Stateful Streaming Processing*. INFOCOM 2023.

task model:



channel queue model
(e.g. $M/M/1$)

service time model

state retrieval time + task (application logic) execution time

cache miss penalty

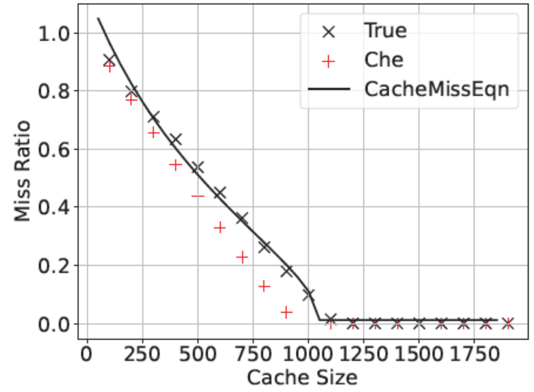
cache miss model

cache miss model:

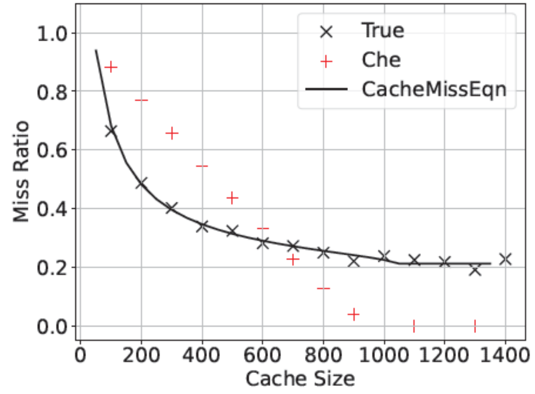
Fagin/Che Approximation

assumptions:

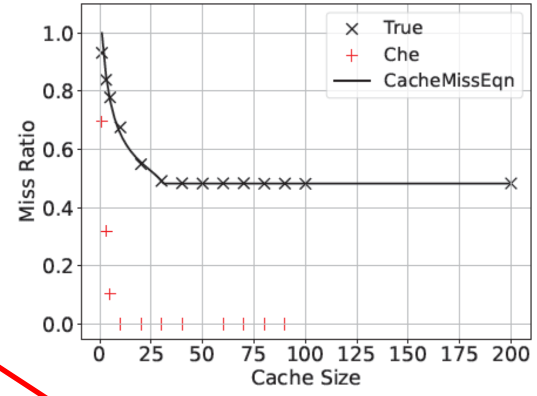
- LRU replacement
- no cold misses, no writes
- uniform object (state) sizes
- independent references (no locality)



(a) Clock caching



(b) A workload with update probability 0.1



(c) Stock trading workload

cache miss model:

Fagin/Che Approximation

assumptions:

LRU replacement

no cold misses, no writes

request rate

characteristic time

hit prob: $h = \sum_n \left(\frac{\lambda_n}{\sum_k \lambda_k}\right) \left(1 - e^{-\lambda_n T_C} \right)$

object

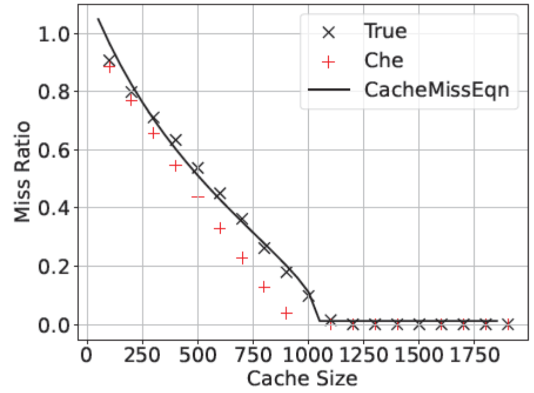
popularity

T_C: $C = \sum_n \left(1 - e^{-\lambda_n T_C} \right)$

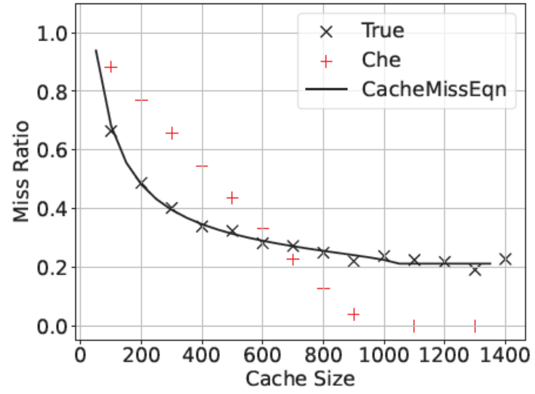
cache size
allocated to task

uniform object (state) sizes

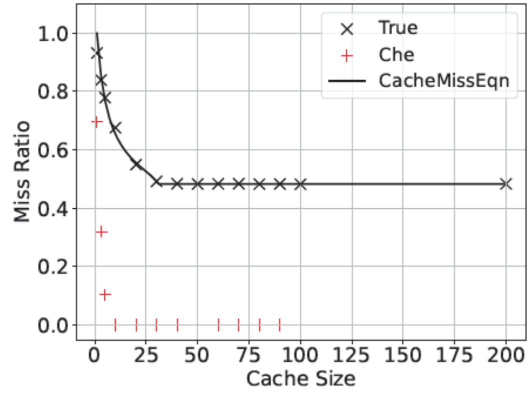
independent references (no locality)



(a) Clock caching



(b) A workload with update probability 0.1



(c) Stock trading workload

cache miss model:

Fagin/Che Approximation

assumptions:

LRU replacement

~~no cold misses, no writes~~

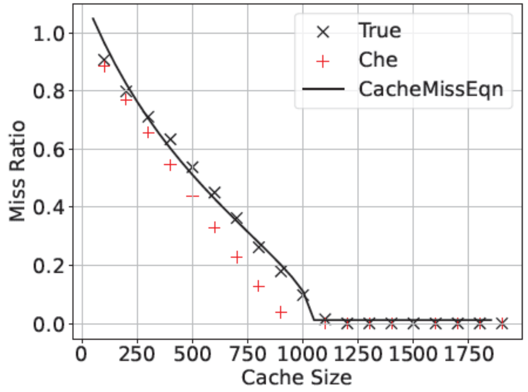
cold miss prob P^*

hit prob: $$h = (1-P^*)\sum_n \left(\frac{\lambda_n}{\sum_k \lambda_k}\right)\left(1 - e^{-\lambda_n T_C}\right)$$

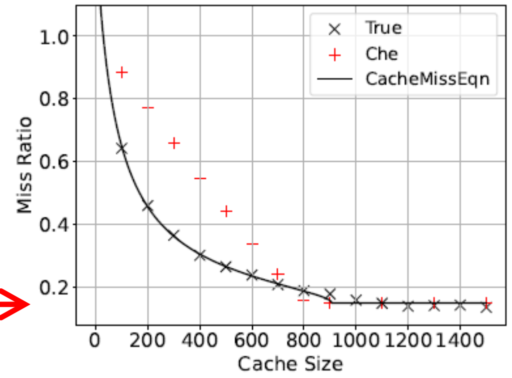
T_C: $$C = (1-P^*)\sum_n \left(1 - e^{-\lambda_n T_C}\right)$$

uniform object (state) sizes

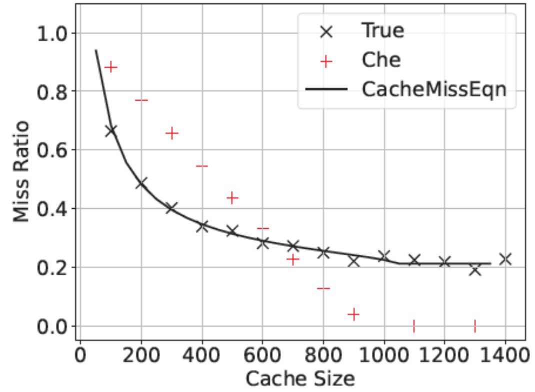
independent references (no locality)



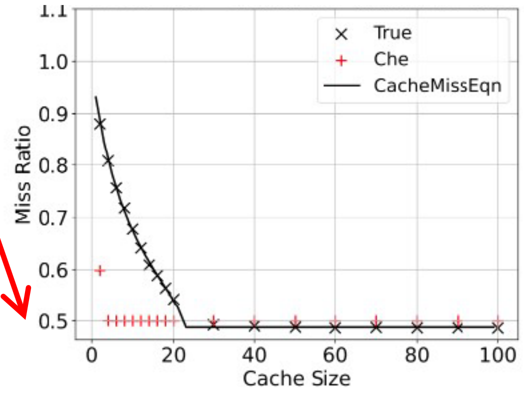
(a) Clock caching



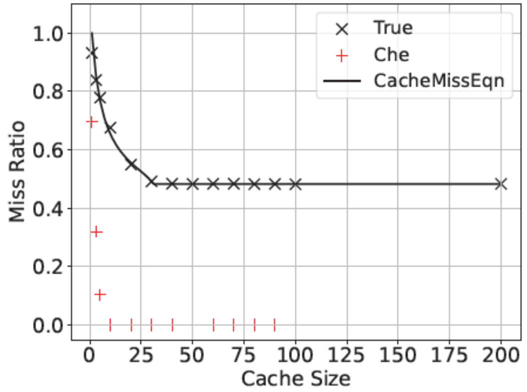
(a) A workload with update probability 0.1



(b) A workload with update probability 0.1



(b) Stock trading workload



(c) Stock trading workload

cache miss model:

Cache Miss Equation

assumptions:

~~LRU replacement~~

~~no cold misses, no writes~~

~~uniform object (state) sizes~~

~~independent references (no locality)~~

cold miss

$P^{miss} = f\,(M \mid M^*, M_b, P^*, P_c)$

dynamic allocation, etc.

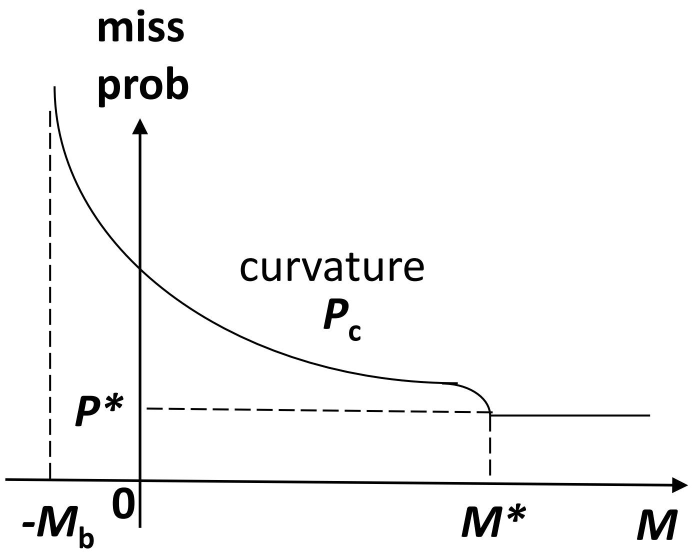
$$= \frac{1}{2}(H + \sqrt{H^2 - 4})(P^* + P_c) - P_c,$$

ideal M

$$\text{where } H = 1 + \frac{M^* + M_b}{M + M_b}, \quad \text{for } M \leq M^*.$$

cache size

space overhead



M^*, M_b , P^*, P_c calibrated by (M, P^{miss}) sample

cache miss model:

Cache Miss Equation

assumptions:

- ~~LRU replacement~~
- ~~no cold misses, no writes~~
- ~~uniform object (state) sizes~~
- ~~independent references (no locality)~~

cold miss

$$P^{\text{miss}} = f\,(M \mid M^*, M_b, P^*, P_c)$$

dynamic allocation, etc.

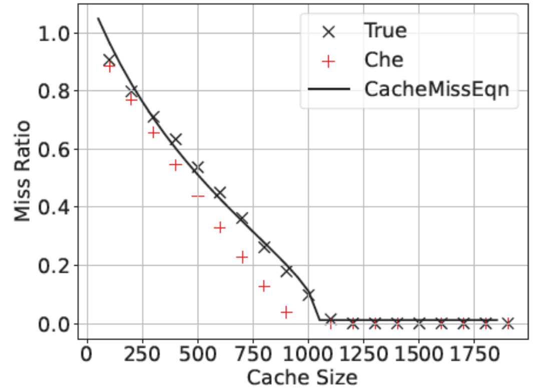
$$= \frac{1}{2}(H + \sqrt{H^2 - 4})(P^* + P_c) - P_c,$$

ideal M

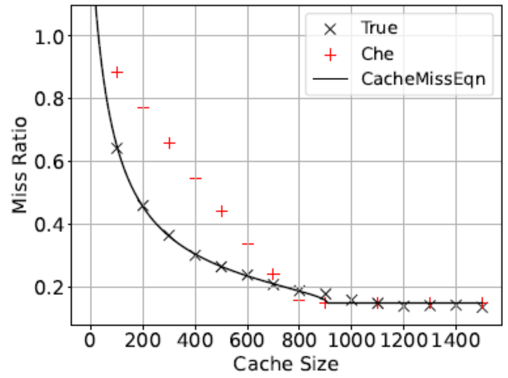
$$\text{where } H = 1 + \frac{M^* + M_b}{M + M_b}, \quad \text{for } M \leq M^*.$$

space overhead

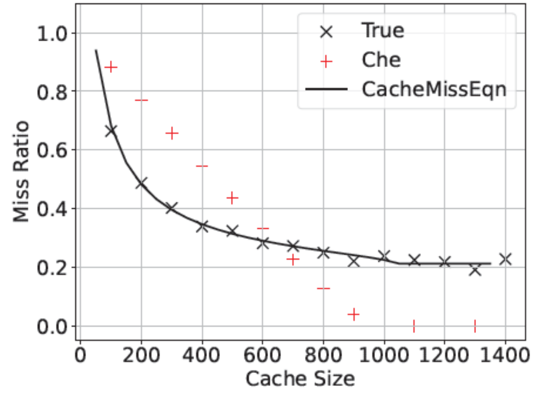
cache size



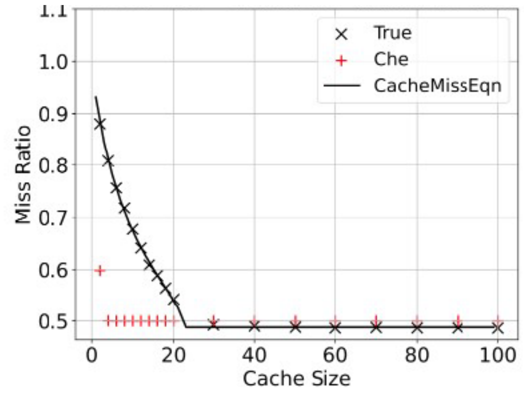
(a) Clock caching



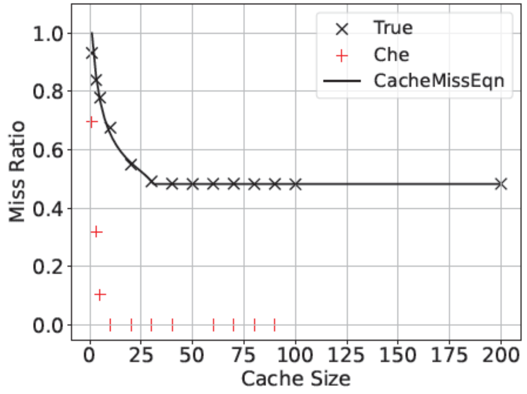
(a) A workload with update probability 0.1



(b) A workload with update probability 0.1



(b) Stock trading workload



(c) Stock trading workload

Cache Miss Equation:

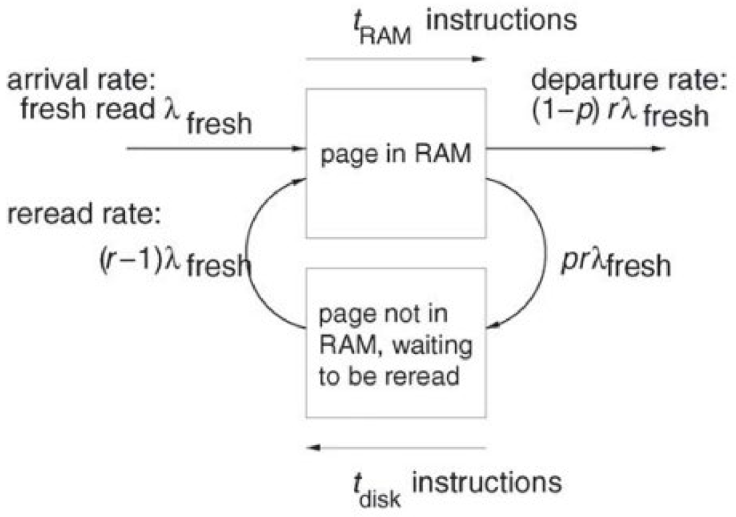
origin:

Y. C. Tay, Min Zou:
A page fault equation for modeling the effect of memory size.
Performance Evaluation (2006).

derivation:

References + Replacement Invariant: For any terminating workload,

$$\left(1-\frac{1}{r}\right)\left(1+\frac{t_{\mathrm{RAM}}}{t_{\mathrm{disk}}}\right)\approx 1 \quad \text{for small } M.$$



Cache Miss Equation

previous applications:

* fair page allocation to processes
  [Tay, Zou: *A page fault equation for modeling the effect of memory size*. Performance Evaluation (2006)]

* tuning database record buffers for a transaction mix
  [Tran, Huynh, Tay, Tung: *A new approach to dynamic self-tuning of database buffers.* ACM ToS (May 2008)]

* sizing heaps for garbage-collected languages
  [Tay, Zong, He: *An equation-based heap sizing rule.* Performance Evaluation 70, 11 (Nov. 2013)]

* partitioning router buffers for Named Data Networking
  [Rezazad, Tay: *A cache miss equation for partitioning an NDN content store.* Proc. AINTEC (2013)]

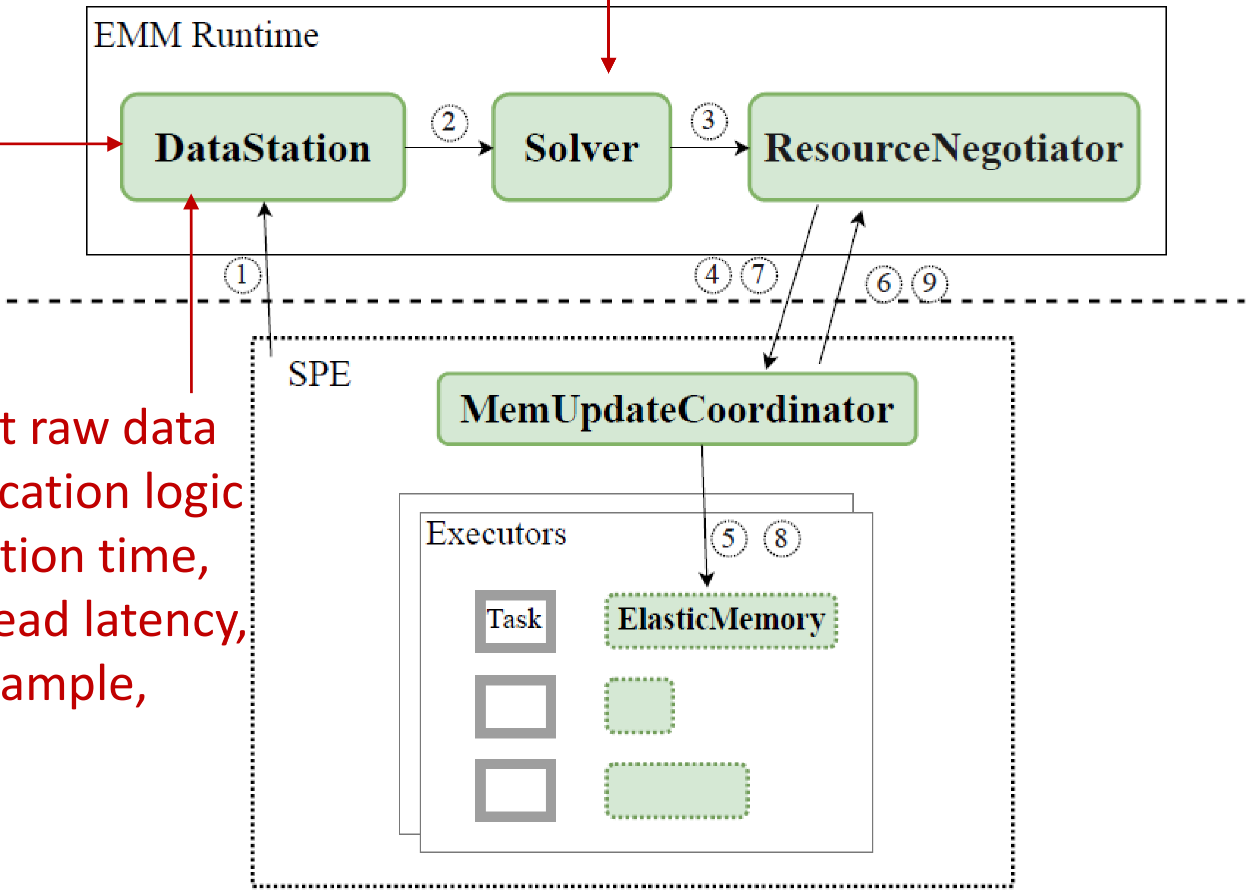
* sizing a 3-level cache
  [Venkatesan, Tay, Zhang, Wei: *A 3-level cache miss model for a nonvolatile extension to transcendent memory.* CloudCom (2014)]

Elastic Memory Manager (EMM):

min max latency

calibrate

M^*, M_b, P^*, P_c



collect raw data (application logic execution time, disk read latency, P^{miss} sample, etc.)

Fig. 4: EMM Framework

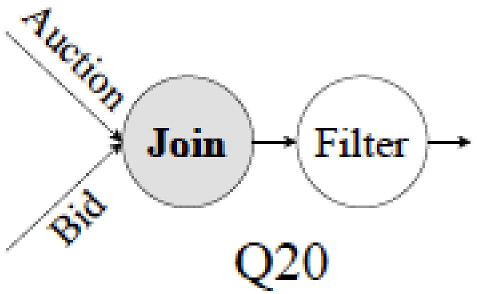
MemUpdateCoordinator embedded into Flink's JobManager

states stored in RocksDB bckend

Elastic Memory Manager (EMM):

using M^* from Cache Miss Equation to allocate memory to each task

NEXMark benchmark query Q20



Join SQL:

```
FROM
    bid AS B INNER JOIN auction AS A on B.auction = A.id
```

Join Operator:

For **Auction**, record in state (writes, 10KB/state), 3000/sec, 48K auctions

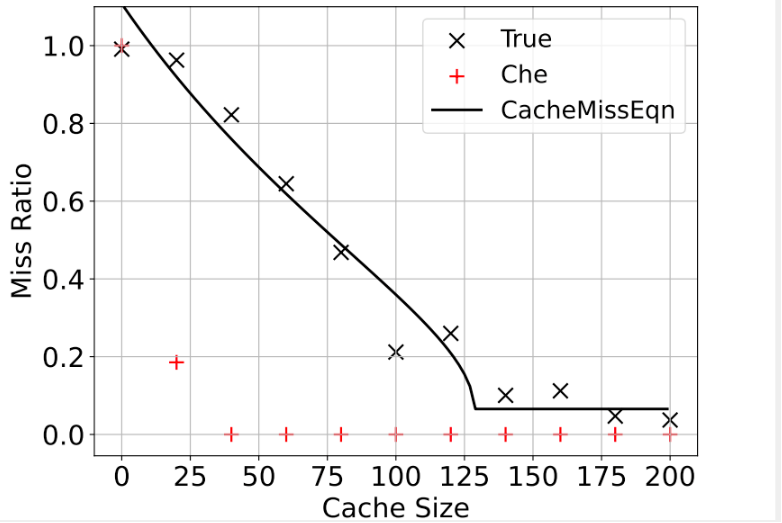
For **Bid**, match the state (random reads of last 10sec), 800/sec

4 tasks (480MB state size, task0 delayed to simulate latency imbalance)

Elastic Memory Manager (EMM):

using M^* from Cache Miss Equation to allocate memory to each task

NEXMark benchmark query Q20



4 Join tasks share 100MB

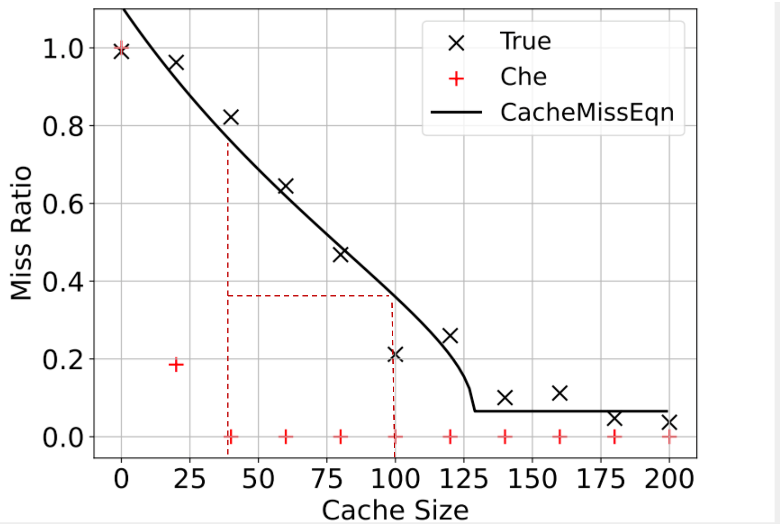
equal share of Bid requests

- task0 slower (EMM: task0 requires $P^{miss} = 0$)

Elastic Memory Manager (EMM):

using M^* from Cache Miss Equation to allocate memory to each task

NEXMark benchmark query Q20



4 Join tasks share 100MB

equal share of Bid requests

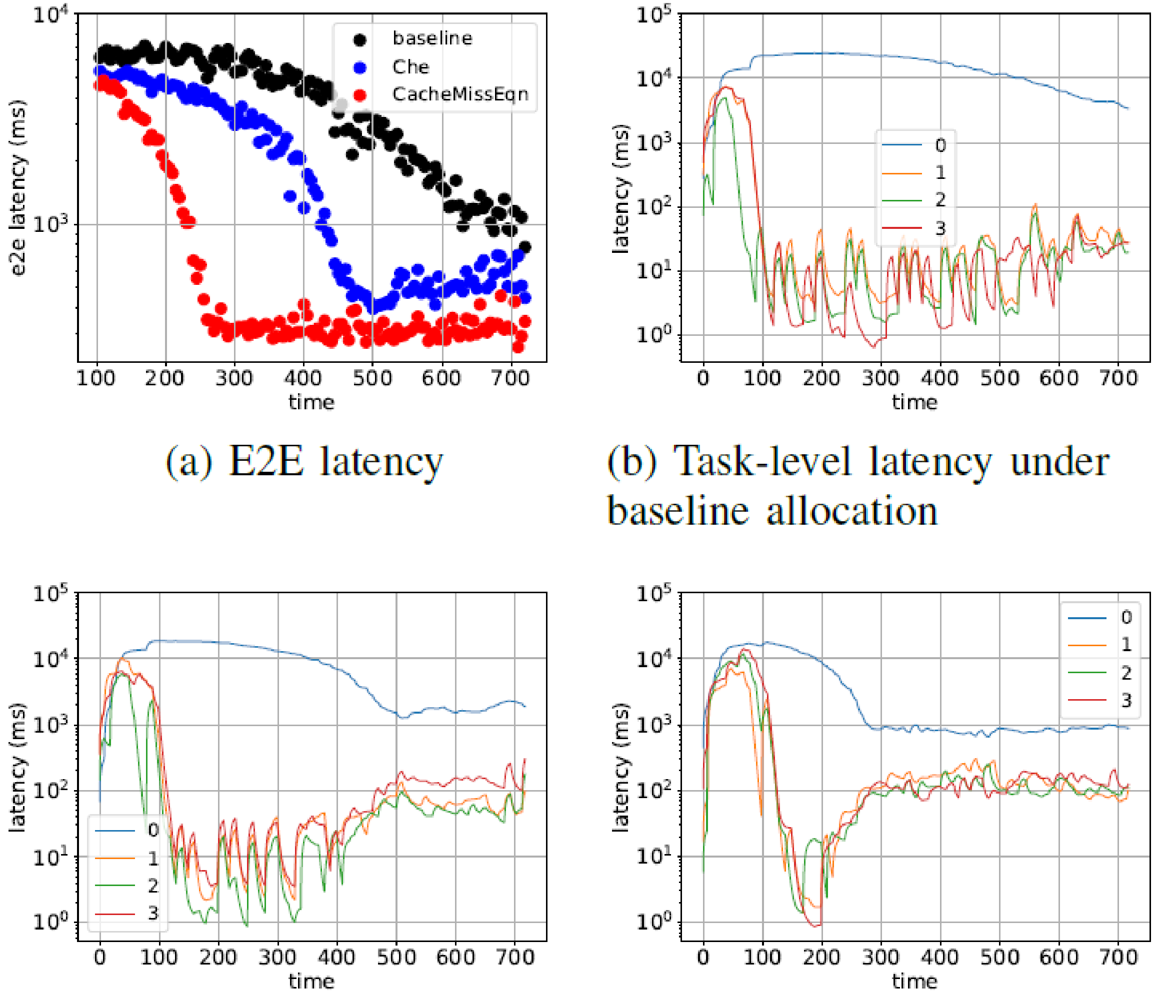
task0 slower (EMM: task0 requires $P^{miss} = 0$)

memory allocation:

baseline: (25MB, 25MB, 25MB, 25MB)

Che: (40MB, 20MB, 20MB, 20MB)

CacheMissEqn: (100MB, 0MB, 0MB, 0MB)



(a) E2E latency

(b) Task-level latency under baseline allocation

(c) Task-level latency under Che's model

(d) Task-level latency under CacheMissEqn model

conclusion:

Fagin/Che Approximation

may not be good for your application

assumptions:

- LRU replacement
- no cold misses, no writes
- uniform object (state) sizes
- independent references (no locality)

request rate

characteristic time

hit prob: $h = \sum_n \left(\frac{\lambda_n}{\sum_k \lambda_k}\right)\left(1 - e^{-\lambda_n T_C}\right)$

object

popularity

T_C : $C = \sum_n \left(1 - e^{-\lambda_n T_C}\right)$

cache size
allocated to task

Cache Miss Equation

may be a better choice

assumptions:

- ~~LRU replacement~~
- ~~no cold misses, no writes~~
- ~~uniform object (state) sizes~~
- ~~independent references (no locality)~~

cold miss

dynamic
allocation, etc.

$$P^{miss} = f\,(M \mid M^*, M_b, P^*, P_c)$$

$$= \frac{1}{2}(H + \sqrt{H^2 - 4})(P^* + P_c) - P_c,$$

ideal M

$$\text{where } H = 1 + \frac{M^* + M_b}{M + M_b}, \quad \text{for } M \le M^*.$$

cache size

space overhead

Dynamic equation-based memory allocation for stream processing engines

Rengan Dou, Richard T.B. Ma, Y.C. Tay
National University of Singapore

$$\#\text{miss} = \frac{1}{2}\left(K + \sqrt{K^2 - 4}\right)(n^* + n_0) - n_0 \quad \text{where} \quad K = 1 + \frac{M^* - M_0}{M - M_0}$$

intuition:

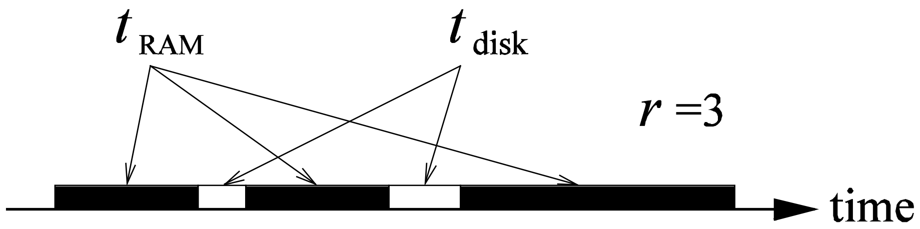
t_{RAM} = average time between entry into RAM and eviction

t_{disk} = average time between eviction and return to RAM

r = average #times a page is read from disk

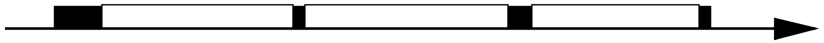
References+Replacement Invariant: $\left(1 - \frac{1}{r}\right)\left(1 + \frac{t_{RAM}}{t_{disk}}\right) \approx 1$

+ Little's Law



likely scenario: small r and large t_{RAM} / t_{disk}

likely scenario: large r and small t_{RAM} / t_{disk}



unlikely scenario: small r and small t_{RAM} / t_{disk}

unlikely scenario: large r and large t_{RAM} / t_{disk}

Tay and Zou, *A page fault equation for modeling the effect of memory size*, Perf. Eval. 2006