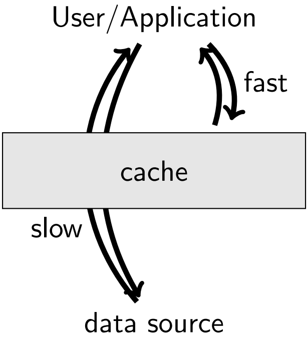
Transient and Steady-state Regime of a Family of List-based Cache Replacement Algorithms

Nicolas Gast[1], Benny Van Houdt[2]

Sigmetrics 2015, Portland, Oregon

[1]Inria
[2]University of Antwerp

Caches are everywhere



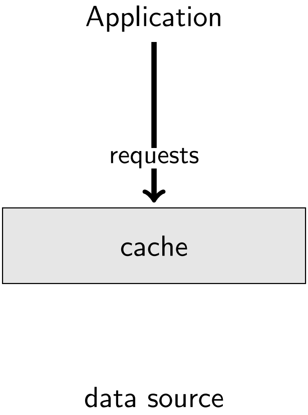
Examples:

- Processor
- Database
- CDN

- Single cache / hierarchy of caches

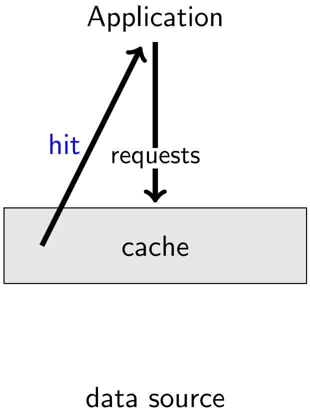
In this talk, I focus on a single cache.

The question is: which item to replace?



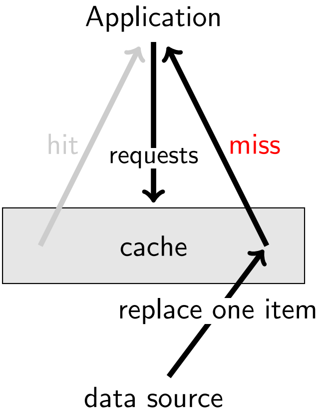
In this talk, I focus on a single cache.

The question is: which item to replace?



In this talk, I focus on a single cache.

The question is: which item to replace?



Classical cache replacement policies:

- RAND, FIFO
- LRU
- CLIMB

Other approaches:

- Time to live

Our performance metric will be the hit probability

$$\text{hit probability} = \frac{\textsf{number of items served from cache}}{\textsf{total number of items served}}$$
$$= 1 - \textsf{miss probability}$$

Our performance metric will be the hit probability

$$\text{hit probability} = \frac{\text{number of items served from cache}}{\text{total number of items served}}$$
$$= 1 - \text{miss probability}$$

- Theoretical studies: IRM (started with [King 1971, Gelenbe 1973])
- Practical studies use trace-based simulations.

- Approximations: link between TTL and cache replacement policies.
  - FIFO and LRU: [Dan and Towsley 1990, Martina at al. 14, Fofack at al. 13, Berger et al. 14]
  - LRU: Che approximation [Che, 2002, Fricker et al. 12]

Contributions (and Outline of the talk)

We introduce a family of policies for which the cache is (virtually) divided into lists (generalization of FIFO/RANDOM)

Contributions (and Outline of the talk)

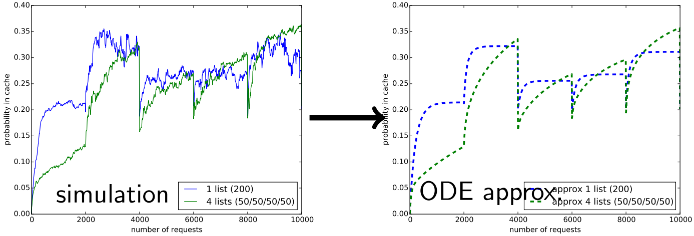
We introduce a family of policies for which the cache is (virtually) divided into lists (generalization of FIFO/RANDOM)

1. We can compute in polynomial time the steady-state distribution under the IRM model.
   - Disprove old conjectures.

Contributions (and Outline of the talk)

We introduce a family of policies for which the cache is (virtually) divided into lists (generalization of FIFO/RANDOM)

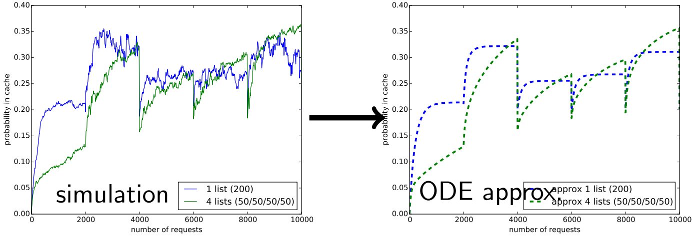
1. We can compute in polynomial time the steady-state distribution under the IRM model.
   - Disprove old conjectures.
2. We develop a mean-field approximation and show that it is accurate
   - Fast approximation of the steady-state distribution.
   - We can characterize the **transient behavior**:



Contributions (and Outline of the talk)

We introduce a family of policies for which the cache is (virtually) divided into lists (generalization of FIFO/RANDOM)

1. We can compute in polynomial time the steady-state distribution under the IRM model.
   - Disprove old conjectures.
2. We develop a mean-field approximation and show that it is accurate
   - Fast approximation of the steady-state distribution.
   - We can characterize the **transient behavior**:



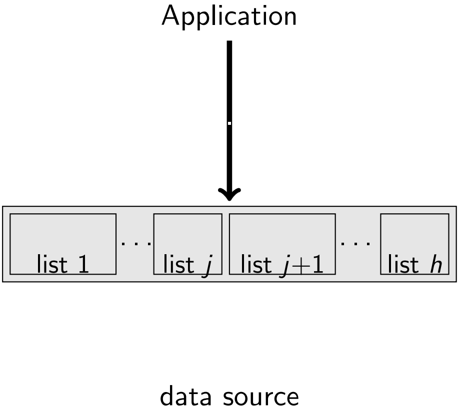
3. We provide guidelines of how to tune the parameters by using IRM and trace-based simulation

Outline

1. Cache model and IRM
2. Steady-state performance under the IRM model
3. Fast and accurate mean-field approximation
4. How to choose the size of the lists?
5. Conclusion

Outline

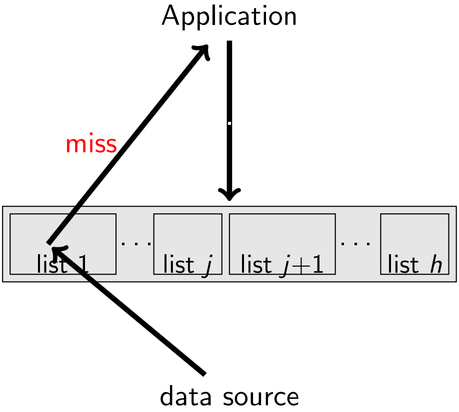
I consider a cache (virtually) divided into lists



IRM At each time step, item i is requested with probability p_i (IRM assumption[3])

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In INFOCOM'99, volume 1, pages 126-134. IEEE, 1999.

I consider a cache (virtually) divided into lists

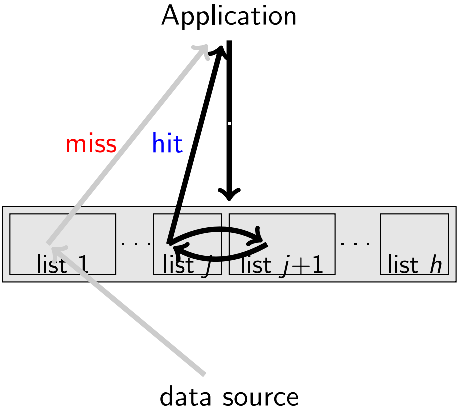


IRM At each time step, item i is requested with probability p_i (IRM assumption[3])

MISS If item i is not in the cache, it is exchanged with a item from list 1 (FIFO or RAND).

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In INFOCOM'99, volume 1, pages 126-134. IEEE, 1999.

I consider a cache (virtually) divided into lists



IRM At each time step, item i is requested with probability p_i (IRM assumption[3])

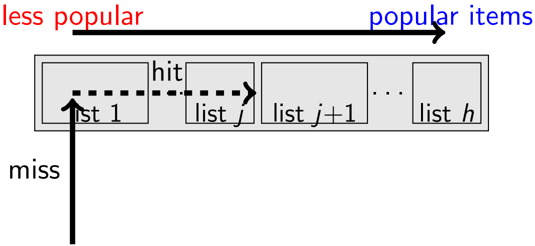
MISS If item i is not in the cache, it is exchanged with a item from list 1 (FIFO or RAND).

HIT If item i is list j, it is exchanged with a item from list $j+1$ (FIFO or RAND).

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In INFOCOM'99, volume 1, pages 126-134. IEEE, 1999.

Items on higher lists are (supposedly) more popular.

$$\text{cache size} = m = m_1 + \cdots + m_h$$



These algorithms are refered to as RAND($\mathbf{m}$)and FIFO($\mathbf{m}$).

Outline

The steady-state is a product-form distribution

THEOREM 1. *The steady state probabilities* $\pi_{RAND(\mathbf{m})}(\mathbf{c})$ *and* $\pi_{FIFO(\mathbf{m})}(\mathbf{c})$, *with* $\mathbf{c} \in \mathcal{C}_n(m)$, *can be written as*

$$\pi_{FIFO(\mathbf{m})}(\mathbf{c}) = \pi_{RAND(\mathbf{m})}(\mathbf{c}) =$$

$$\pi(\mathbf{c}) \triangleq \frac{1}{Z(\mathbf{m})} \prod_{i=1}^{h} \left(\prod_{j=1}^{m_i} p_{c(i,j)} \right)^i, \qquad (1)$$

where $Z(\mathbf{m}) = \sum_{\mathbf{c} \in \mathcal{C}_n(m)} \prod_{i=1}^{h} \left(\prod_{j=1}^{m_i} p_{c(i,j)} \right)^i$.

- Same for RAND and FIFO.

The steady-state is a product-form distribution

THEOREM 1. *The steady state probabilities* $\pi_{RAND(\mathbf{m})}(\mathbf{c})$ *and* $\pi_{FIFO(\mathbf{m})}(\mathbf{c})$*, with* $\mathbf{c} \in \mathcal{C}_n(m)$*, can be written as*

$$\pi_{FIFO(\mathbf{m})}(\mathbf{c}) = \pi_{RAND(\mathbf{m})}(\mathbf{c}) =$$

$$\pi(\mathbf{c}) \triangleq \frac{1}{Z(\mathbf{m})} \prod_{i=1}^{h} \left(\prod_{j=1}^{m_i} p_{c(i,j)} \right)^i, \qquad (1)$$

where $Z(\mathbf{m}) = \sum_{\mathbf{c} \in \mathcal{C}_n(m)} \prod_{i=1}^{h} \left(\prod_{j=1}^{m_i} p_{c(i,j)} \right)^i$.

- Same for RAND and FIFO.

Example of a cache of size 4 with 3 lists and $\mathbf{m} = (1, 2, 1)$:

Probability of (i, j, k, ℓ) is proportional to $p_i(p_j p_k)^2(p_\ell)^3$.

We can compute the miss probability by using a dynamic programming approach (Generalization of [Fagin,Price][5]).

We want to compute

$$M(\mathbf{m}) = \sum_{\mathbf{c}\in\mathcal{C}_n(m)} \left(\sum_{k\notin\mathbf{c}} p_k\right) \pi(\mathbf{c}) = \frac{E(\mathbf{m}+\mathbf{e}_1, n)}{E(\mathbf{m}, n)},$$

where $E(\mathbf{r}, k) = \sum_{\mathbf{c}\in\mathcal{C}_k(r)} \prod_{i=1}^{h} \left(\prod_{j=1}^{r_i} p_{c(i,j)}\right)^i.$

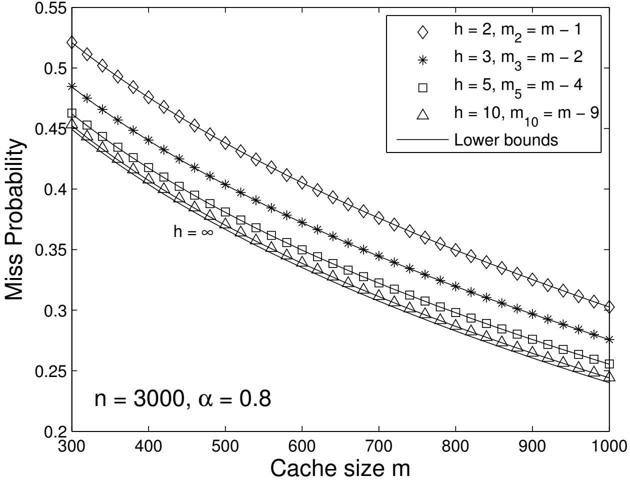
We obtain a recursion formula on $E(\mathbf{r}, k)$: solvable in $O(n \times m_1 \dots m_h)$.

The Dan and Towsley[4] approximation is not needed for polynomial time.

[4] A. Dan and D. Towsley. An approximate analysis of the LRU and FIFO buffer replacement schemes. SIGMETRICS Perform. Eval. Rev., 18(1):143-152, Apr. 1990.

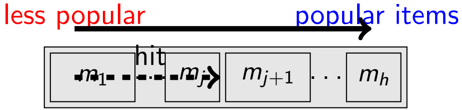
[5] R. Fagin and T. G. Price. Efficient calculation of expected miss ratios in the independent reference model. SIAM J. Comput., 7:288-296, 1978.

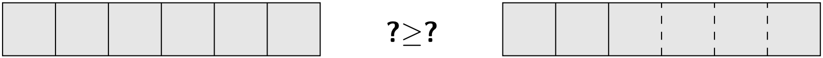
A higher cache size and more lists (usually) leads to a lower steady-state miss probability.



($h = \infty$ corresponds to LFU).

Is increasing the number of lists always better[6]?





Six lists: $\mathbf{m} = (1, 1, 1, 1, 1, 1)$

Three lists: $\mathbf{m} = (1, 1, 4)$.

[6] conjectured in O. I. Aven, E. G. Coffman, Jr., and Y. A. Kogan. Stochastic Analysis of Computer Storage. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

Is increasing the number of lists always better[6]?

Six lists: $\mathbf{m} = (1,1,1,1,1,1)$

$?\geq?$

Three lists: $\mathbf{m} = (1,1,4)$.

policy	m	$M(\mathbf{m})$	lower bound
Optimal	RAND(1,1,4)	0.005284	0.004925
	RAND(1,1,3,1)	0.005299	0.004884
	RAND(1,1,2,2)	0.005317	0.004884
	RAND(1,1,2,1,1)	0.005321	0.004879
	RAND(1,1,1,3)	0.005338	0.004884
	RAND(1,1,1,2,1)	0.005343	0.004879
	RAND(1,1,1,1,2)	0.005347	0.004879
CLIMB	RAND(1,1,1,1,1,1)	0.005348	0.004878
	RAND(1,2,3)	0.005428	0.004925
	RAND(1,2,2,1)	0.005439	0.004884
LRU	LRU(6)	0.005880	–
RANDOM	RAND(6)	0.015350	0.015350

Table 1: CLIMB is not optimal for IRM model: $p = (49,49,49,49,7,1,1)/205$ **and** $m = 6$**.**

[6] conjectured in O. I. Aven, E. G. Coffman, Jr., and Y. A. Kogan. Stochastic Analysis of Computer Storage. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

Outline

We want to study at which speed the caches fills

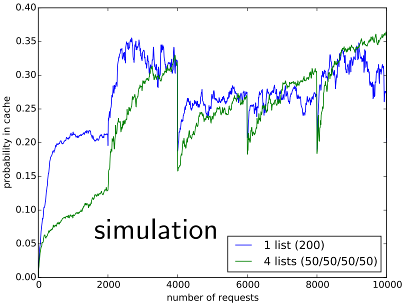


Figure : Popularities of objects change every 2000 steps.

We want to study at which speed the caches fills

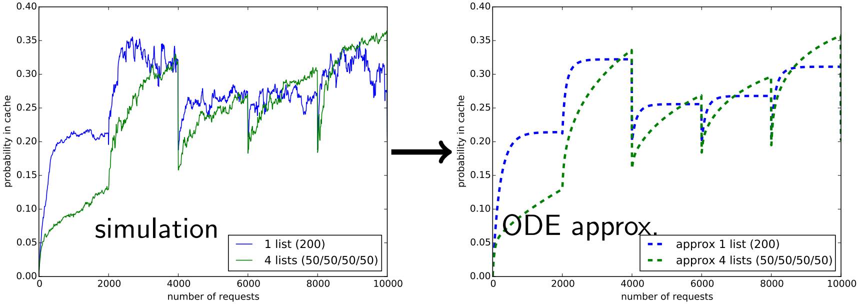


Figure : Popularities of objects change every 2000 steps.

- We develop an ODE approximation

We want to study at which speed the caches fills

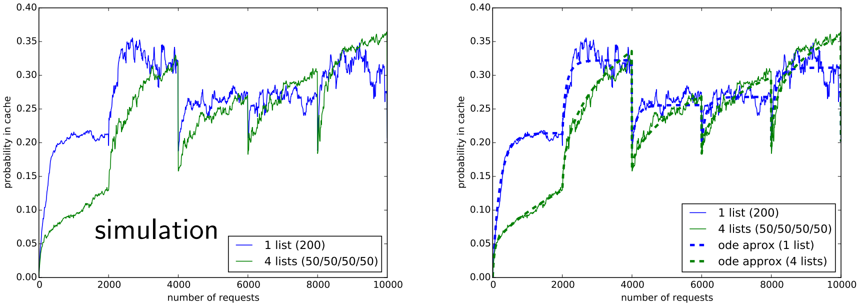
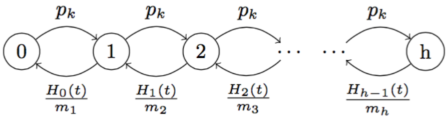


Figure : Popularities of objects change every 2000 steps.

- We develop an ODE approximation
- We show that it is accurate

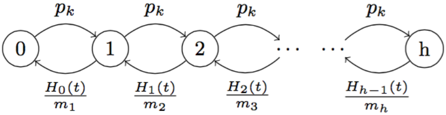
We construct an ODE by assuming independence

Let $H_i(t)$ be the popularity in list i.



We construct an ODE by assuming independence

Let $H_i(t)$ be the popularity in list i.



If $x_{k,i}(t)$ is the probability that item k is in list i at time t, we approximately have:

Popularity in cache i-1

$$\dot{x}_{k,i}(t) = p_k x_{k,i-1}(t) - \sum_j p_j x_{j,i-1}(t) \frac{x_{k,i}(t)}{m_i} + \mathbf{1}_{\{i<h\}} \left(\sum_j p_j x_{j,i}(t) \frac{x_{k,i+1}(t)}{m_{i+1}} - p_k x_{k,i}(t) \right)$$

Popularity in cache i

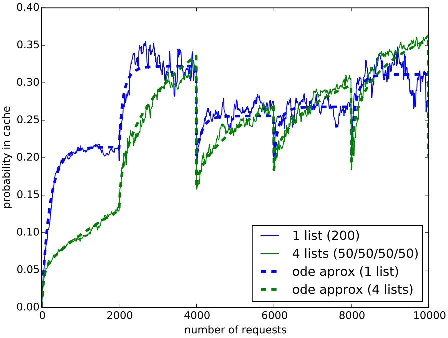
This is similar to a TTL approximation.

We show that this approximation is accurate, theoretically and by simulation

THEOREM 6. *For any $T > 0$, there exists a constant $C > 0$ that depends on T such that, for any probability distribution over n items and list sizes $m_1 \dots m_h$, we have:*

$$\mathbf{E}\left[\sup_{t\in\{0\dots\tau\},i\in\{0\dots h\}} |H_i(t)-\delta_i(t)|\right] \leq C\sqrt{\max_{k=1}^{n} p_k + \max_{i=0}^{h} \frac{1}{m_i}},$$

where $\tau := \lceil T/(\max_{k=1}^{n} p_k + \max_{i=0}^{h} \frac{1}{m_i})\rceil$.



This approximation can also be used to compute stationary distribution

THEOREM 7. *The mean-field model (8) has a unique fixed point. For this fixed point, the probability that item k is part of list i, for $k = 1, \dots, n$ and $i = 0, \dots, h$, is given by*

$$x_{k,i} = \frac{p_k^i z_i}{1 + \sum_{j=1}^{h} p_k^j z_j},$$

where $\mathbf{z} = (z_1, \dots, z_h)$ is the unique solution of the equation

$$\sum_{k=1}^{n} \frac{p_k^i z_i}{1 + \sum_{j=1}^{h} p_k^j z_j} = m_i. \qquad (14)$$

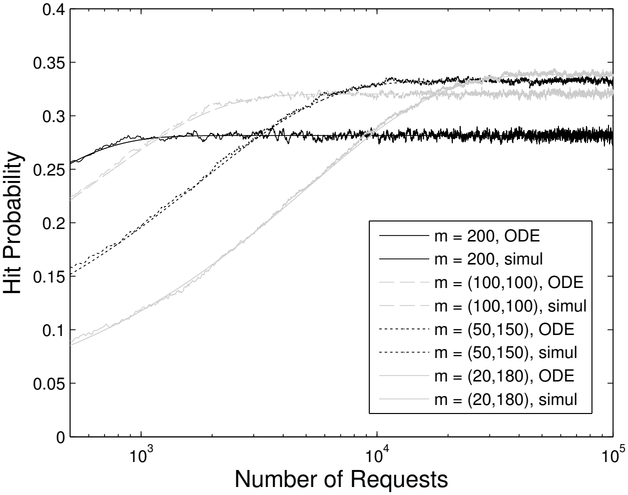
- Very accurate:

m_1	m_2	m_3	m_4	exact	mean field
2	2	96	–	0.3166	0.3169
10	30	60	–	0.3296	0.3299
20	2	78	–	0.3273	0.3276
90	8	2	–	0.4094	0.4100
1	4	10	85	0.3039	0.3041
5	15	25	55	0.3136	0.3139
25	25	25	25	0.3345	0.3348
60	2	2	36	0.3514	0.3517

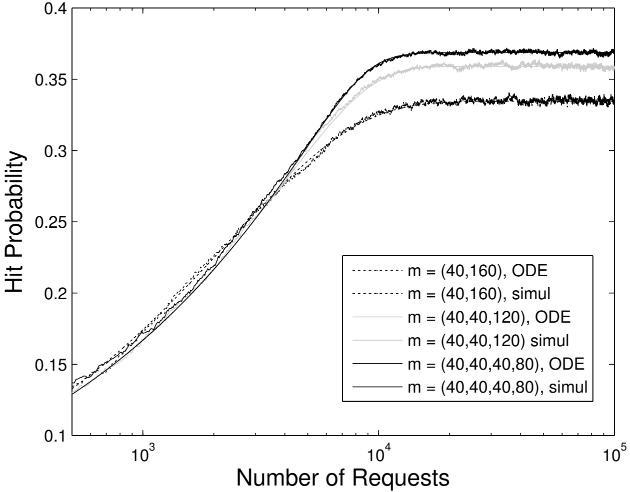
- Map is contracting: computation in $O(nh)$, compared to $O(nm_1 \dots m_h)$ for the exact.

Outline

Under the IRM model, a smaller first list (usually) means a higher hit probability but a larger time to fill the cache

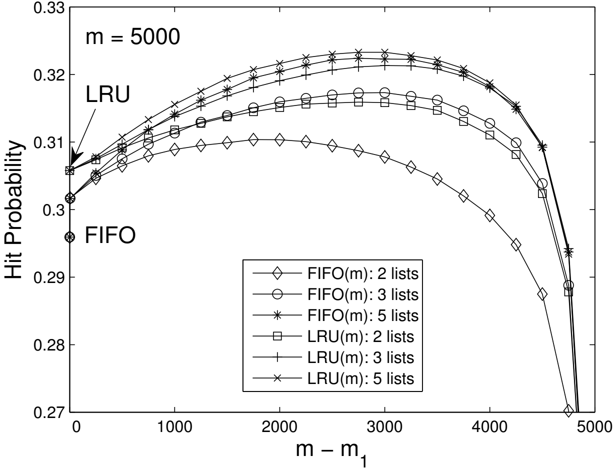


Under the IRM model, the time to fill the cache mainly depend on the size of the first list.



- In a dynamic setting, a good choice seems to be $m_1 \geq m_2 \cdots \geq m_h$ with m_1 “large-enough”.

We verified on a trace of youtube videos[7], that reserving at least 30% of the cache for the first list seems important.

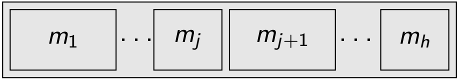


[7] M. Zink, K. Suh, Y. Gu, and J. Kurose. Characteristics of YouTube network traffic at a campus network-measurements, models, and implications. Comput. Netw., 53(4):501-514, Mar. 2009.

Outline

Conclusion

- Unified framework for studying list-based replacement policies.
- Steady-state miss probability in polynomial time.
- Accurate ODE approximation
- Guidelines on how to use such a replacement algorithm: the size of the first list is important.



- Two theoretical interests of this work:
  - provides a unified framework and disproves old conjectures.
  - ODE approximation
- Future work: network of caches.

Thank you!

`http://mescal.imag.fr/membres/nicolas.gast`

`nicolas.gast@inria.fr`

Transient and Steady-state Regime of a Family of List-based Cache Replacement Algorithms.