The Price of Forgetting in Parallel Routing

Jonatha Anselmi & Bruno Gaujal

INRIA

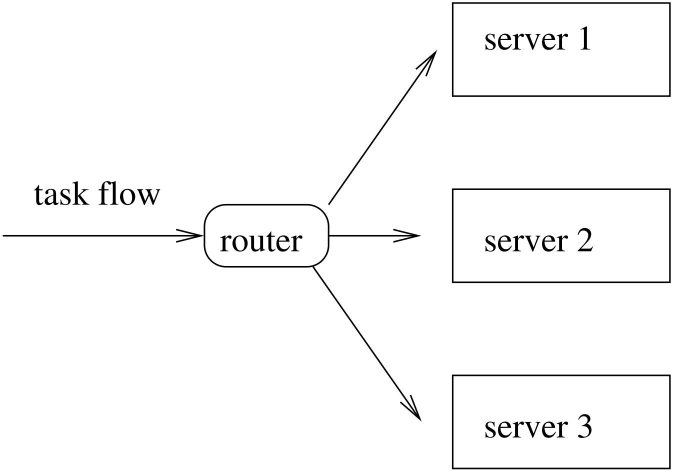
Aussois – June, 2011

Introduction

Goal of my talk : explore different types of routing policies (selfish/social, with/without memory) in a task-resource system and compare their performances.

This leads to the introduction of the concepts of price of anarchy and price of forgetting, respectively.

Flow in parallel servers



Three models of information structure

- Complete information : Instance (packet sizes and arrival times) is fully known by the router.
- No information : The instance is completely unknown to the router that only discovers the data as it comes.
- Statistical information : The router does not know the actual instance but has some knowledge about its statistics (arrival rate, average size of packets, distribution,...)

Two models for the cost

The cost model can either be

- the worse case : the worse possible response time over all tasks (WCET),
- or the average case : the mean response times over the set of tasks, equipped with a distribution.

On-Line vs Off-line Scheduling

Here, the controller has statistical information on the instance x (arrrival times and sizes) and minimizes the average response time $\mathbb{E}(r_\pi(x))$.
Off-Line case : the controller must take all its decisions beforehand.
On-Line case : the controller sees the current state (backlog) up to time n and can adapt its decisions to it,
(they coincide in the deterministic case).
The expected cost of the optimal policy at time n is :
Off line :

$$\inf_{a_1,\cdots,a_n} \mathbb{E}(r_{a_1,\cdots,a_n}(x)).$$

On Line :

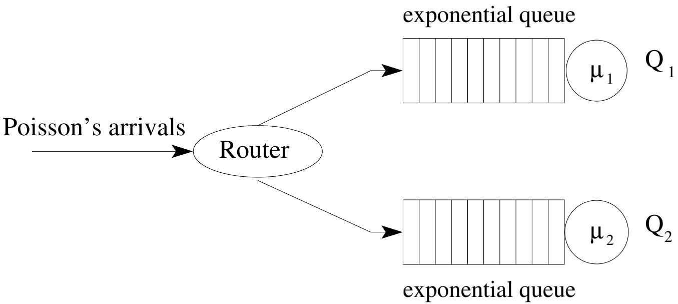
$$\inf_{d_1,\cdots,d_n} \mathbb{E}(r_{d_1(x_1),\cdots,d_n(x_n)}(x)).$$

Theorem

There exists an optimal deterministic policy (in both cases).

Average response time in parallel queues

Assumptions on the arrival times and task sizes.



Service times in queues serve packets at rate μ_1 and μ_2 resp.
The arrival sequence is Poisson with parameter λ.

On-Line : Optimal Control Problem

This problem can be solved numerically in the on-line case using optimal control techniques.
The computation is NP-hard in general (with m servers).

Theorem

When the servers are identical, Join the Shortest Queue (Selfish policy) (JSQ) is an optimal policy.

Theorem (Weber, R. and Weiss, G. (1990))

When the number of servers goes to infinity, index policies are optimal.

Off-line : Bernoulli Policy

As for off-line policies, the scheduler has to decide where to send each job in advance.

One possibility : send jobs to queues with probabilities $p_1, \cdots, p_m$.

The optimal Bernoulli policy can be computed using the following mathematical program.

$$R^{Opt}_{Bernoulli} = \min_{p_1, \dots p_m} \sum_{i=1}^{m} \frac{p_i}{\mu_i - \lambda p_i}$$

under the constraints $\sum_i p_i = 1$ and $0 \leq p_i < \mu_i / \lambda$.

This problem can be solved in closed form using a Lagrangian relaxation.

Off-line : Bernoulli Policy

The i_s fastest servers are used, where

$$i_s = \min \left\{ i \geq 1 : \mu_{i+1} \leq \frac{(\mu_{(i)} - \lambda)^2}{(\sum_{j=1}^{i} \sqrt{\mu_j})^2} \right\}.$$

Moreover, the optimal probability p_i^* to chose server $i \leq i_s$ is

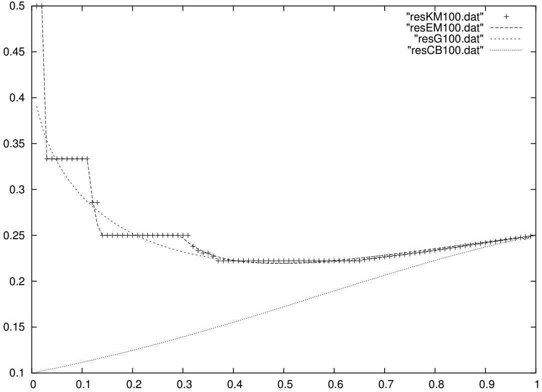
$$p_i^* = \frac{1}{\lambda}(\mu_i - \frac{\sqrt{\mu_i}}{\beta})$$

where $\beta \stackrel{\text{def}}{=} \frac{\sum_{j=1}^{i_s} \sqrt{\mu_j}}{\mu_{(i_s)} - \lambda}$. Finally, the mean response time in the utilized server i is

$$R_{Bernoulli}^{Opt} = \beta \sqrt{\mu_i}, \quad i \leq i \leq i_s.$$

More advanced off-line policies

Here, we compare with Gamma and with a mixture of Erlangs (where the optimal solution can also be computed).



Price of Forgetting

Price of Forgetting measures the benefit of having memory in the scheduler.

$$PoF \stackrel{\text{def}}{=} R^{Opt}_{Bernoulli}/R^{Opt}. \tag{1}$$

Computing R^{Opt} in the off-line case is very difficult (open problem). There exists non trivial lower bounds :

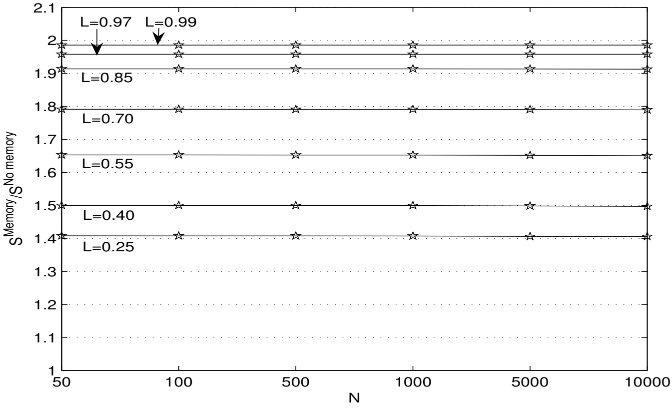
$$R^{Opt} \geq \inf_{\substack{p_1,\dots,p_N \geq 0: \\ p_1+\cdots+p_N=1}} \sum_{i=1}^{N} p_i R_i^{D(\lambda/p_i)/GI/1}.$$

Theorem

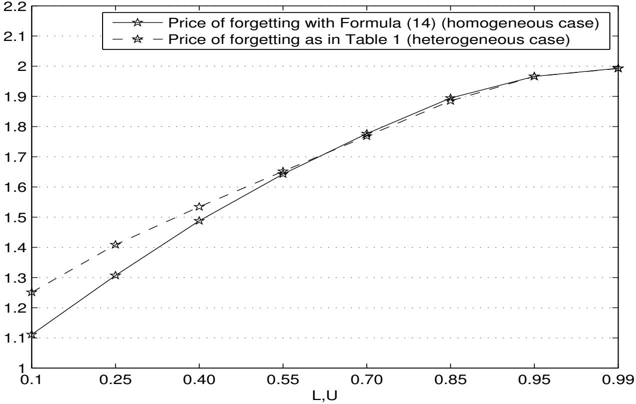
$$PoF(N) \leq 1 + \frac{1}{\min_{i=1}^{N} \mu_i^2 s_i^2}.$$

The PoF is bounded by 2 in the exponential case (but can be unbounded when the coefficient of variation goes to 0).

Price of Forgetting (II)



Price of Forgetting (III)



Price of Anarchy

The Price of Anarchy (PoA) [Papadimitriou, 99] is an index measuring the inefficiency of a decentralized system with respect to its centralized counterpart in presence of selfish users.
Here, it is the response-time ratio between the worst-case situation where each task is selfish (maximizes its own response time) and the contrasting situation where jobs are routed optimally by a scheduler, yielding the *social optimum*.

$$PoA(N) \stackrel{\text{def}}{=} \frac{R^{We}(N)}{R^{Opt}(N)} \geq 1.$$

Selfish routing

Tasks wish to minimize their mean waiting time and select a server accordingly. They are allowed to randomize regarding their choice of servers.
The solution is a symmetric Nash equilibrium under steady-state conditions :

The waiting times in all used servers are equal.

The k fastest servers are used : $k = \min\left\{1 \le i \le N : \mu_{i+1} \le \frac{\mu_{(i)} - \lambda}{i}\right\}$.

The probability to join server i is

$$\overline{p}_i = \frac{1}{\lambda}\left(\mu_i - \frac{k}{\mu_{(k)} - \lambda}\right).$$

and the corresponding response time is

$$R^{We}(N) = \frac{k}{\mu_{(k)} - \lambda}.$$

The Price of Anarchy with a twist

The performance ratio between the selfing routing using probabilities $(\overline{p}_1, \ldots, \overline{p}_N)$ and the best routing probabilities $(p_1^*, \ldots, p_N^*)$ is

Theorem (Haviv and Roughgarden, 2007)

$$R^{We}(N)/R^{opt}_{Bernoulli}(N) \leq N \qquad (tight)$$

The Price of Anarchy with a twist

The performance ratio between the selfing routing using probabilities $(\overline{p}_1, \ldots, \overline{p}_N)$ and the best routing probabilities $(p_1^*, \ldots, p_N^*)$ is

Theorem (Haviv and Roughgarden, 2007)

$$R^{We}(N)/R^{opt}_{Bernoulli}(N) \leq N \qquad \textit{(tight)}$$

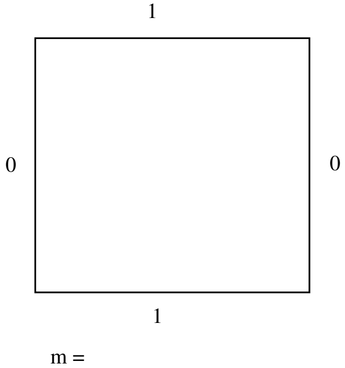
$$PoA(N) = PoA_{Bernoulli}(N) PoF(N).$$

In the exponential case, since $PoF(N) \leq 2$, $PoA(N) \leq 2N$.

Optimal Policy : Billiard sequences

Theorem

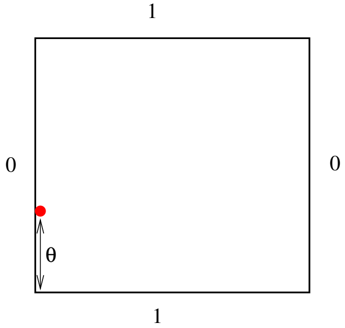
Billiard sequences are optimal routing policies in two queues (or N deterministic fully loaded queues). They perform within 1% of optimal in most cases.



Optimal Policy : Billiard sequences

Theorem

Billiard sequences are optimal routing policies in two queues (or N deterministic fully loaded queues). They perform within 1% of optimal in most cases.

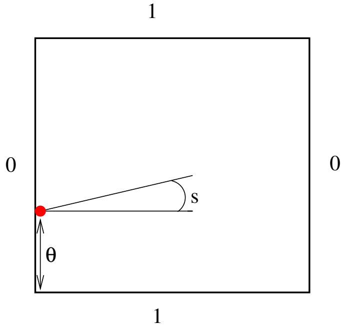


m =

Optimal Policy : Billiard sequences

Theorem

Billiard sequences are optimal routing policies in two queues (or N deterministic fully loaded queues). They perform within 1% of optimal in most cases.

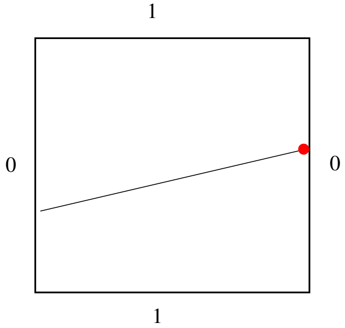


m =

Optimal Policy : Billiard sequences

Theorem

Billiard sequences are optimal routing policies in two queues (or N deterministic fully loaded queues). They perform within 1% of optimal in most cases.

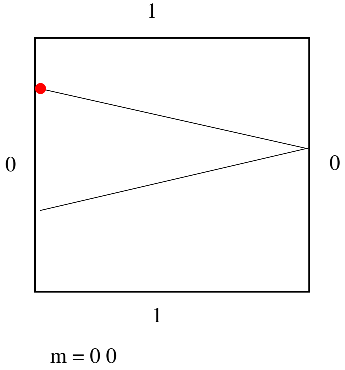


m = 0

Optimal Policy : Billiard sequences

Theorem

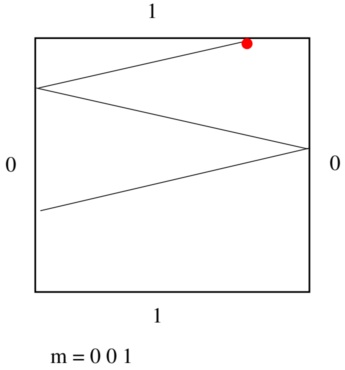
Billiard sequences are optimal routing policies in two queues (or N deterministic fully loaded queues). They perform within 1% of optimal in most cases.



Optimal Policy : Billiard sequences

Theorem

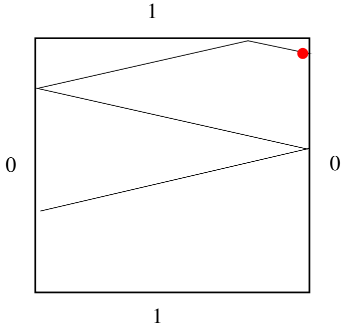
Billiard sequences are optimal routing policies in two queues (or N deterministic fully loaded queues). They perform within 1% of optimal in most cases.



Optimal Policy : Billiard sequences

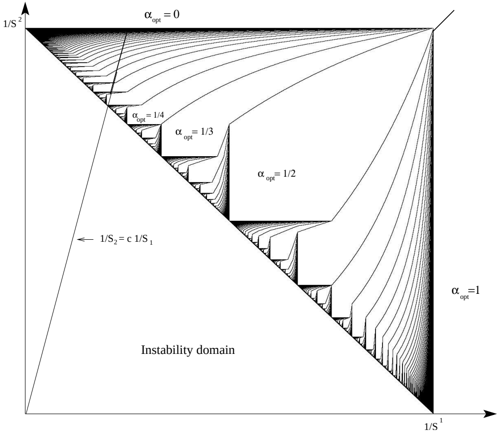
Theorem

Billiard sequences are optimal routing policies in two queues (or N deterministic fully loaded queues). They perform within 1% of optimal in most cases.



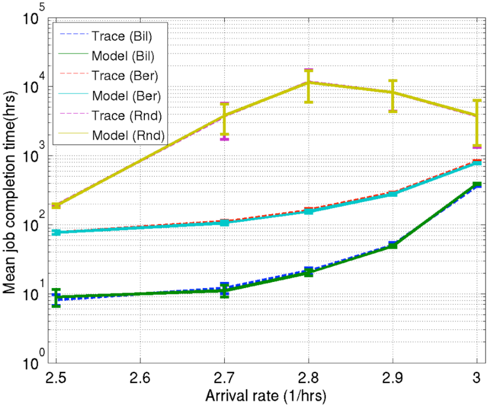
m = 0 0 1 0

The hard part : Rate computation



Real life test

Billiard sequences have been tested in a Boinc application (from D. Kondo and B. Javadi).



Thank you