Replication in Computer Systems - DM

Jonatha Anselmi

February 4, 2020

> Votre devoir sera rédigé en français ou en anglais sous forme d'un document HTML généré à l'aide de R/Markdown et publié sur rpubs en prenant soin de bien laisser le code apparent et de fixer la graine de votre générateur à l'aide de la fonction set.seed au tout début du document afin qu'il soit possible de reproduire vos données avec exactitude.
> Vous enverrez l'url rpubs de votre devoir par mail à arnaud.legrand AT imag.fr et à jonatha.anselmi AT inria.fr en indiquant dans le sujet [RICM4-EP] DM avant le 12 mars à minuit.

Introduction

Nowadays, it is crucial that computer systems respond to interactive users in a few milliseconds. In practice, user requests, or jobs in the following, are subject to a number of factors that increase the variability of response times. These include unfortunate disk seek times, background daemons and run-time contention phenomena among CPU cores, processor caches, memory bandwidth, and network bandwidth. As a result, some jobs may take significantly longer than expected to complete while keeping blocked resources that could be used by other concurrent interactive jobs. This has brought researchers to propose hedge requests, or redundant jobs in the following. The idea consists in *replicating a job upon its arrival and use the results from whichever replica responds first*. In other words, when a new job arrives, it is replicated to a given number of different servers for processing and as soon as one replica completes or starts service, the response is sent back to the issuing user and the other replicas are canceled. Redundant requests are used by Google's big table services [1].

[1] J. Dean and L. A. Barroso. *"The Tail at Scale."* Commun. ACM 56, 2 (Feb. 2013), 74–80.

Homework assignment

The goal of this homework assignment is i) to write a code in R that simulates the dynamics of redundant jobs in a multiserver platform and ii) to use the code to answer some practical questions. We consider a system of K parallel servers (or queues) adopting the first-come first-served discipline. When a new job arrives, it is replicated to d queues selected uniformly at random *without replacement* and independently of all else. We assume that the service times of all replicas of all jobs are i.i.d. random variables having the distribution of the random variable S (specified later). We assume that the jobs' interarrival times are independent and have an exponential distribution with rate λ, i.e., jobs join the system following a Poisson process with rate λ.

Depending on the assumptions on S, we are interested in understanding which one of the following two scenarios is the most convenient:

Scenario 1: the remaining copies of each job are canceled as soon as any copy *starts* service at some server;
Scenario 2: the remaining copies of each job are canceled as soon as any copy *completes* service at some server.

Step 1: Simulation of redundant jobs

Modify the code that simulates the dynamics of a G/G/1 queue (http://polaris.imag.fr/arnaud.legrand/teaching/2019/RICM4_EP.php#orge330de7) to simulate K G/G/1 queues as described above.

System parameters

- `N` : number of arriving jobs to simulate (excluding replicas);
- `d` : number of replicas per job;
- `K` : number of queues;
- `lambda` : job arrival rate;
- `mu` : in the case where S follows an exponential distribution, $\mu = 1/\mathbb{E}[S]$ is the job service rate.
- `xm,alpha` : in the case where S follows a Pareto distribution (https://en.wikipedia.org/wiki/Pareto_distribution), `xm` resp. `alpha` is the scale resp. shape parameter. Since common values of `alpha` found in the empirical studies of computer systems range in the interval $[1, 2]$, we assume `alpha=1.5` .

Using the inversion method, note that

```
xm=1;
alpha=1.5;
xm/(runif(5)^(1/alpha));
```

```
## [1]   1.431324   1.406144   1.823874 136.823037   1.230806
```

gives a sample associated to 5 i.i.d. Pareto distributed random variables.

State variables

- `t` : simulation time;
- `Arrival` : size-N vector of jobs' arrival times;
- `Service` : size-$N \times d$ matrix of jobs' service times;
- `Queue` : size-$N \times d$ matrix indicating where the i-th copy of the n-th job is dispatched;
- `Remaining` : size-$N \times d$ matrix indicating the remaining time of the i-th copy of the n-th job;
- `Completion` : size-N vector indicating the job's completion time;
- `CurrJob` : size-K vector indicating the current job in processing at each server, i.e., `CurrJob[i]` takes value in $\{1, \ldots, N\}$;
- `CurrReplica` : size-K vector indicating the current replica in processing at each server, i.e., `CurrReplica[i]` takes value in $\{1, \ldots, d\}$;
- `NextJob` : this variable is incremented by one each time a new job arrives (not necessary but simplifies the code).

Do not hesitate to introduce other variables if needed.

Code structure (for both scenarios 1 and 2)

```
while (TRUE) {
    dtA = ... # time of the next arrival
    dtC = ... # time of the next completion
    if(is.na(dtA) & is.na(dtC)) {break;}
    dt = min(dtA,dtC)

    if((NextArrival <=N) & (Arrival[NextArrival] == t))
    {
      # update Remaining and, possibly, and other state variables
    }

    for(k in 1:K)
    {
      if(!is.na(CurrJob[k]))
      {
        # update Remaining and, possibly, CurrJob to NA and other state vari
ables
      }
    }

    for(k in 1:K)
    {
      if(is.na(CurrJob[k]))
      {
        # assign a job and a copy to server k
        # update state variables
      }
    }

}
```

Note for the evaluation: The more efficient the code, the better. Efficiency refers to the ability of simulating an increased number of jobs given a time constraint.

Step 2: Numerical investigation

We call *response time* the average time that jobs spend in the network. Answer the following items distinguishing between the cases where S follows an exponential or Pareto distribution.

Question 1

Assume that `d=1` and that S has an exponential distribution. Give a formula for the response time.

Question 2

Assume that `N=1e4` or higher, `d=4` , `K=10` and $\mathbb{E}[S] = 1$. Then,

1. Plot the response time as a function of `lambda` , where `lambda` varies between 0 and $0.95K/\mathbb{E}[S]$, and verify that it corresponds to the formula given in previous question when `d=1` and S has an exponential distribution.
2. Is `N` large enough to make the previous plot representative of the mean time that jobs spend in the network?

Question 3

Assume that `N=1e4` or higher, `K=10` , `lambda=0.7*K` and $\mathbb{E}[S] = 1$. Use your code to investigate under which values of the replication parameter `d` :

1. the system is stable, i.e., the response time does not diverge to infinity as $N \to \infty$.
2. Scenario 1 is better than Scenario 2 (and viceversa).

Explain your numerical investigations providing plots and intuititions that support your conclusions.