NETWORKING FOR BIG DATA
DATA CENTER MODULE

Challenge 2: Job dispatching and scheduling

Floyd
Leonardo Di Nino : 1919479
Gian Alvin Guico : 2033024
Luca Mazzucco : 1997610

May 29, 2023

Defining our dispatching and scheduling policy

First of all we are restricting our field of analysis to head of line scheduling policies and anticipative dispatching procedures, so after we checked that the memory constraint is verified for every task we can forget about it, since processor sharing won't be considered. In particular stating that SRPT is well-known to be optimal [1] in single server queues but its optimality is still an open challenge in multiserver systems[2] we won't necessarily implement it as our scheduling policy. We'll then discuss about whether LWL or SITA better fits with our scenario.

Exploratory data analysis

For the preprocessing of the data we just dropped all the task with zero CPU required and scaled back the arrival time to seconds.

There are some crucial aspects to balance the workload in this system simulation aiming to minimize the mean job response times, but we have to state the two principal approaches we can adopt:

- We can work task-wise, defining a policy that try to avoid long queue for short task;
- We can work job-wise, defining a policy that try to avoid long queue for single-task jobs.

The following task-wise insights on the task size show us some really important points.

Global variance	18794.41339
Range	[9.5367431640625e-07, 52787.07165527344]
Fraction of single-task-job-tasks	0.947832
Fraction of multi-task-job-tasks	0.052168
Single-task job max task running time	52787.071655
Single-task min	0.000001
Inner variance in single-task group	17815.960462
Multi-task job max task running time	20778.893906
Multi-task job min task running time	9.5367431640625e-07
Inner variance in multi-task group	19687.25102688618

We have high variability , but at the same time the two groups are almost homoskedastic: this means that if we work task-wise we can think of the workflow as an homogeneous one with respect to the two groups. Viceversa, working job-wise, clearly the whole thing gets unbalanced, since grouped tasks take way longer to be completed:

Variance of total CPU required	3.737372e+07
Max value for total CPU required	1.184767e+06
Min value for total CPU required	2.765656e-05

So we are going to consider a task-wise approach, and stating the insights this actually favors mostly single-task jobs.

Dispatching policy: LWL or SITA?

Given the high-variability scenario, the idea is to implement SITA policy to keep shorts tasks from getting stuck behind long ones, but the stringent segregation of the groups can lead to underutilization of the servers, and the groups can be unbalanced if the division doesn't take into account the proportions of different sized jobs; more over, we have shown during lectures that SITA and LWL dispatching policy can either converge or diverge under different conditions. So we want to retrieve from this data enough information to choose one of the two policies: in particular, we want to find out if both LWL and SITA may diverge assuming a Pareto

[1] L.Schrage, *A proof of the optimality of the shortest remaining processing time discipline*, in "Operations Research, Vol.16, No.3", (1968), pp.687-690

[2] I.Grosof, Z.Scully, M.Harchol-Balter, *SRPT for multiserver systems*, in "Elsevier:Performance Evaluation 127-128 (2018)", pp.154-175

distribution for the task size: in this case the distribution of job response time under LWL is uniformly upper bounding SITA.

We have an important result coming from [3], where two theorems are stated about bounded Pareto underlying processes: if $X \sim BoundPareto(k, \rho, \alpha)$, we have that given n servers:

- SITA always diverges
- If $\alpha > \frac{3}{2}$ and $\rho < n - 1$, then LWL converges; else LWL diverges.

In this notation X is the job size and ρ is the mean job size: in our case we have that $\rho \approx 7.98 < 63 = n - 1$. Through a repeated subsampling over the whole population and using the distfit module in Python we achieved a maximum likelihood estimator for α in this parametric Paretian hypotesis of $\alpha \approx 1.2$, so we are in the scenario where both LWL and SITA diverges with respect to the job response time as the charge of work increases, and in this case SITA is a better choice.

Customizing SITA: implementing SITA-E and injecting a little of LWL

Stating this we now have to dig deeper into some insights to properly define a way to set thresholds in defining this SITA dispatcher: this is one of the hardest tasks when defining a SITA policy. In order to define the N groups of tasks to be assigned to the N servers it has been shown that the optimal SITA policy for a dispatcher on an homogeneous FCFS farm is SITA-E [4], where the thresholds are set such that the workload is balanced between all of the servers. Given the complexity of the problem, we implemented a very handmade heuristic. We initialized 64 arrays a_i and started populating them from the first one scanning the ordered set S of service times: we only switch to the subsequent array a_{i+1} if $\sum a_i \leq \frac{\sum_j S_j}{64}$. Then we retrieved the maximum value from each of this group and set it as the i-th threshold. What happened is that actually two of our servers, due to the approximation, were not used, so we decided to use them to lighten the workload of the busiest one, implementing a LWL subroutine only for the subsystem given by the servers 0, 62 and 63.

Stating the algorithm

Algorithm 1 SITA-E(LWL) dispatching over a FCFS farm

8

Require: $data : matrix, th : array$
  $clock \leftarrow data[0, "ArrivalTime"]$ ▷ Instant in which the procedure begins
  $q_i \leftarrow [\,],\ i = 0, ..., 63$ ▷ Preallocating a queue for each server
  $U_i \leftarrow 0,\ i = 0, ..., 63$ ▷ Preallocating workload for each queue
  for $h = 1, ..., nrow(data)$ **do**
    $\delta = data[h, "ArrivalTime"] - clock$
    for $j = 1, ..., 64$ **do**
      $U_j \leftarrow max(0, U_{j-1} - \delta)$
      $q_j \leftarrow FCFSscheduling(q_j)$[5]
    end for
    $i \leftarrow \underset{i=1,...,64}{\text{argmax}} \{th[i] \geq data[h, "ServiceTime"]\}$
    if $i = 0$ **then**
      $i \leftarrow \underset{k=0,62,63}{\text{argmin}} \{U_i\}$
    end if
    $q_i \leftarrow append(q_i, data[h,])$
    $U_i \leftarrow U_i + data[h, "ServiceTime"]$
    $clock \leftarrow data[h, "ArrivalTime]$
  end for

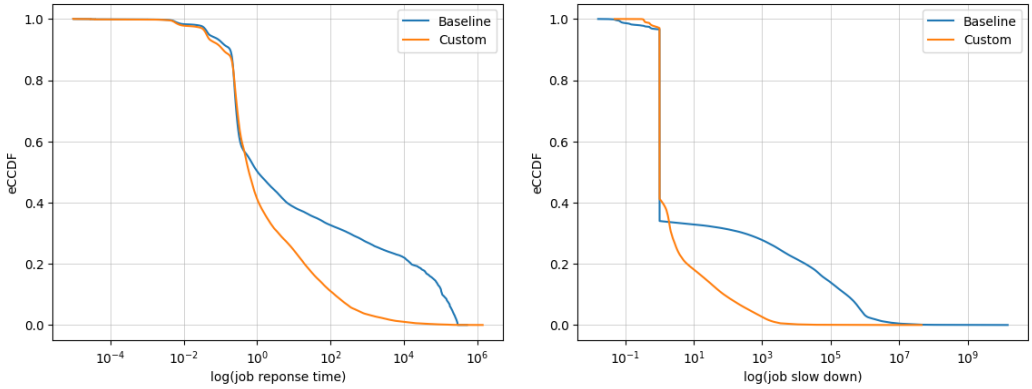
[3] M.Harchol-Balter, A.Scheller-Wolf,A.Young, *Surprising Results on Task Assignment in Server Farms with High-Variability Workloads* , in "SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems", pp.287-298

[4] J.Anselmi,J.Doncel, *Asymptotically Optimal Size-Interval Task Assignments*, in "IEEE Transactions on Parallel and Distributed Systems ", ISSN: 1045-9219

[5] This subroutine just have to track the completion of tasks respecting the FIFO logic

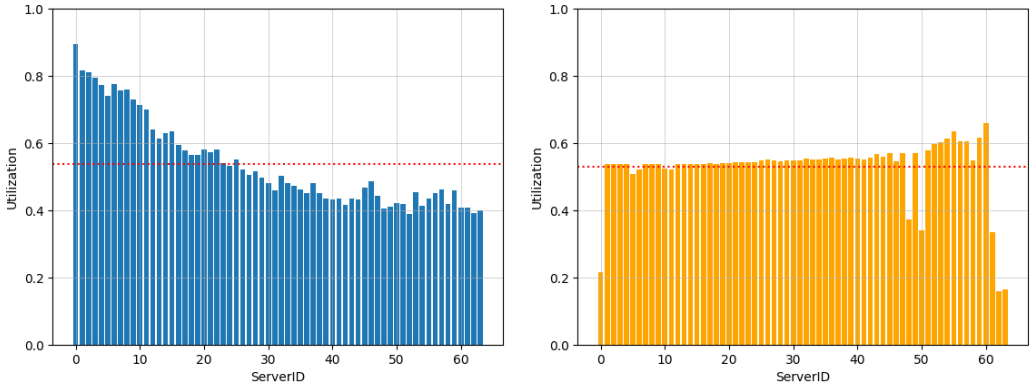
Performance metrics

	Mean Job Response Time	Mean Job Slow Down	Mean utilization	Mean Message Load
Baseline	27603.901135	1.241676e+06	0.538425	129.000000
Custom	739.771959	7.548976e+02	0.528506	5.606374



(a) eCCDF of job response time (b) eCCDF of job slow down

Figure 1: Plotting metrics: job response time and job slow down



(a) Utilization coefficients in baseline implementation (b) Utilization coefficients in custom implementation

Figure 2: Plotting metrics: utilization coefficients

Analysis and comments

- As we can see surely our analysis has been experimentally confirmed: the SITA-E with FCFS servers strongly improved time performances. The right tail of the distributions in the custom case is very weaker.
- For what concerns utilization we didn't achieved any improvement: this is due to the strong balancing we induced that sets every server on the same amount of work time-wise speaking. The sub-LWL we designed over server 0, 62 and 63 surely affects their utilization.
- Message load in the baseline case is a constant for all task, since 128 messages are needed to retrieve the least amount of work left (64 queries and 64 answers), plus 1 message to send the task; in the custom scenario we have to average between pure SITA where we only need 1 message and the sub-LWL where we need 7 messages for task (3 queries over server 0, 62 and 63, 3 answers and 1 task-message).