TFM-FernandoMartín

Fernando Martín Canfrán

January 18, 2025

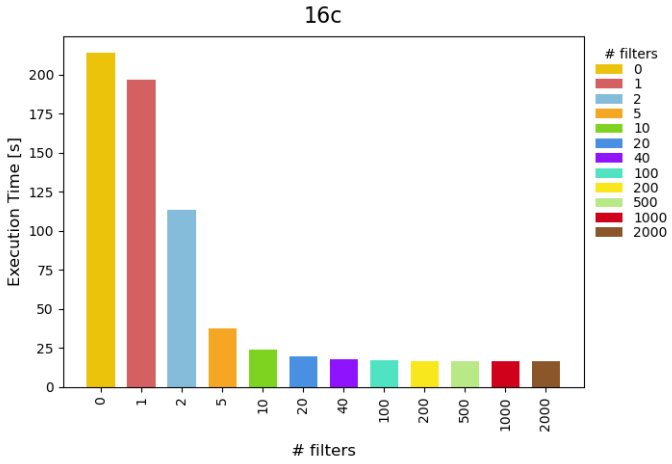
1 Analysis of results

Results Analysis

- Better behavior for a number of filters: 5,10,20 (not lowest, not really high)
- How the number of cores influences in the improve of the behavior

Comparison of behavior depending on the number of filters

Exec time plots - for a core number - different stream sizes



30-0.02

images/4-Experiments/NRT/small

60-0.02

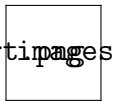
images/4-Experiments/NRT/small/120-0.02/16c/execTim

120-0.02

Figure 1: Execution Time Plots for 16c

MRT plots - for a core number 16 - different stream sizes

30-0.02

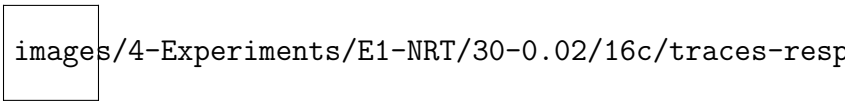


60-0.02

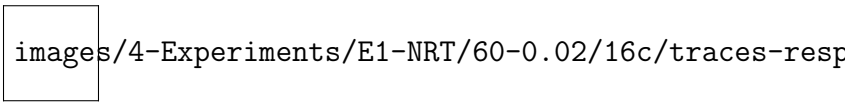
120-0.02

Figure 2: MRT Plots for 16c

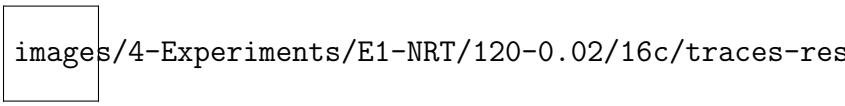
Trace and response time trace for 16c and big stream



(a) 30-0.02

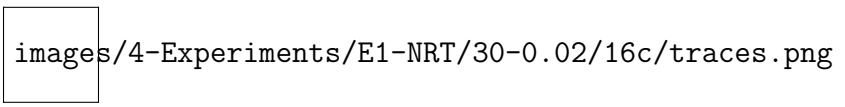


(b) 60-0.02

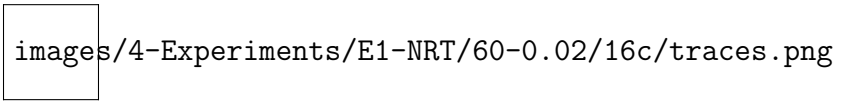


(c) 120-0.02

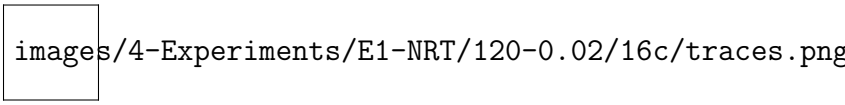
Figure 3: Response Time Traces for 16c



(a) 30-0.02



(b) 60-0.02



(c) 120-0.02

Figure 4: Results Traces for 16c

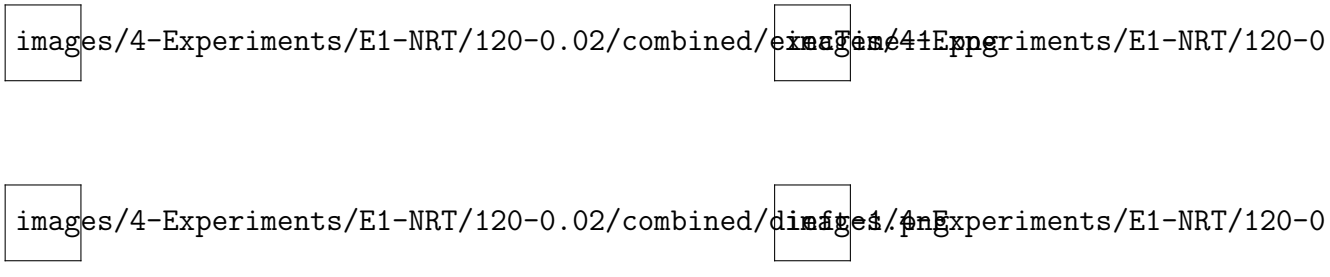


Figure 5: Combined cores and filters variation plots for big stream: 120-0.02

Trace and response time trace for 16c and big stream

Fernando: Decidir qué poner, redactar bien las interpretaciones

Interpretation:

- **??** For the smallest stream size the response time of 1f (sequential) is always higher than when using more filters. However for larger stream sizes this is not the case, since from a certain point the response time tends to be higher, showing an accumulation... overhead?

- Conclusions: less filters maintain constant the RT (especially better for large stream sizes / where we have longer periods of high loaded scenarios). More filters however tend to get increased their response time, accumulating. In the middle, 5, 10 and 20 filters tend to be the best tradeoff between low and constant response time and low execution time / time needed to process all the input stream.

- The continuous behavior is better while increasing the number of filters for all the number of cores variations in terms of the `dief@t` metric, this can be seen in the `dief@t` plot in the Figure **??**. It can also be seen in the traces plot in Figure **??** which shows the result traces in the case of the variations run with 16c. Note the slight degradation of the `dief@t` in the variations with the highest number of filters (from 100 and on, but specially on the cases with 1000 and 2000 filters), in the variations run with low number of cores: Figure **??**.

- However measuring the behavior in terms of the execution time (total needed time to process the full stream input) and the mean response time and response time traces, we can see that for a same core configuration (e.g. 16 cores), the total execution time (time to process all the stream input) is larger for the approaches with less filters, tending to decrease. However the behavior in terms of the response time is different: the best is observed for a number of filters in the range of 5-10 filters. From that point and on the mean response time tends to degrade when increasing the number filters, especially for bigger stream sizes. Even larger than the lowest number of filters version (close to a sequential version) in these cases. This can be due to an overhead on the number of goroutines utilized and the overhead in the communication of the pipeline that this is causing.

Comparison of behavior depending on the number of cores

- More cores help to improve the behavior, especially for the variants with large number of filters (expected).

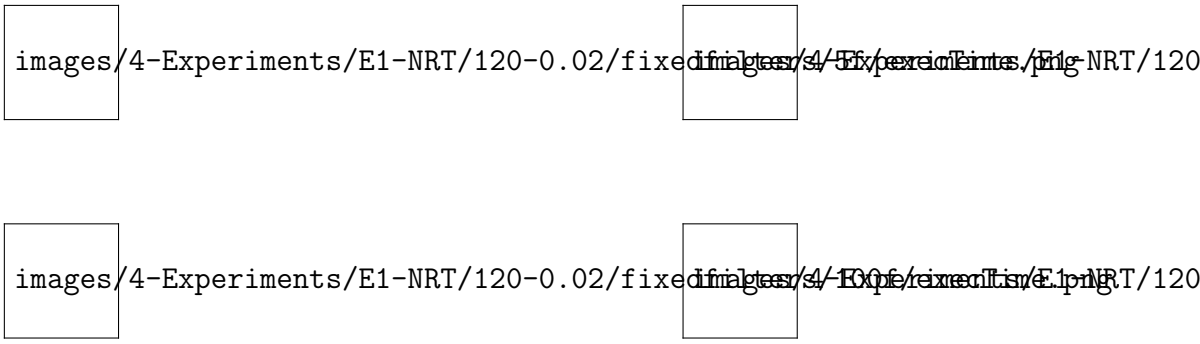


Figure 6: Execution Time Fixed filters plots for stream 120-0.02

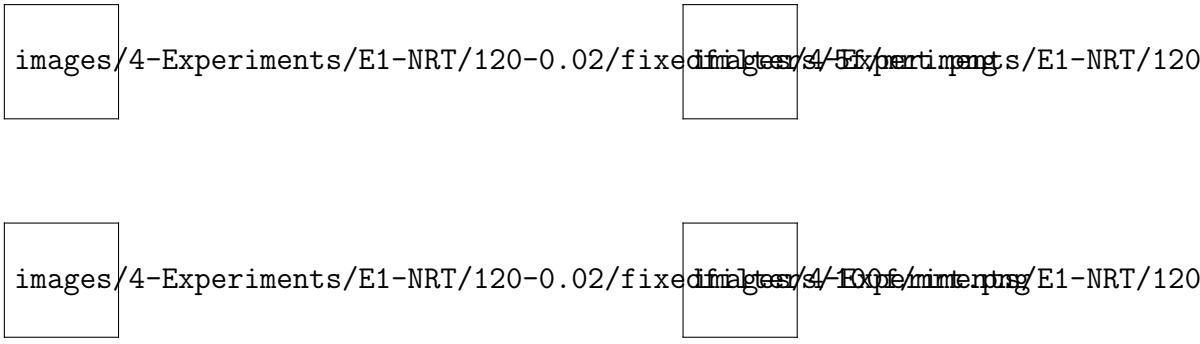


Figure 7: Mean Response Time Fixed filters plots for stream 120-0.02

1.0.1 Medium Bank Size

For these experiments, to generate the stream of tx, we needed to simplify this process in order to be able to generate a stream in a feasible amount of time. In particular we used the simplifed version of the `txGenerator.py`: `txGenerator-simplified.py` → with a random ATM-subset instead of a closest to client ATM-subset. Also variation on the transaction distribution times.
Run with:

- 16GB RAM
- x1 run each job

E2: Evaluation in a Real-World Stress Scenario

Fernando: TODO: Describir y poner resultados

For the experiments perform The real-case scenario and the high loaded test scenario. In the first case, the interactions, although read by a file of artificial simulated interactions, are provided to the pipeline data stream in such a way that they simulate their actual arrival time to the system, with the corresponding time separation

between them. In the second case, the interactions are provided just one after the other as fast as possible as they are read.

Fernando: TODO: Describir la configuración/entorno donde corrimos los experimentos - maquinas del cluster, con + o - RAM... poner sus características