A Scheduling Framework for Distributed Key-Value Stores and Application to Tail Latency Minimization

S. Ben Mokhtar[1], L.-C. Canon[2], A. Dugois[3], L. Marchal[3] and E. Rivière[4]

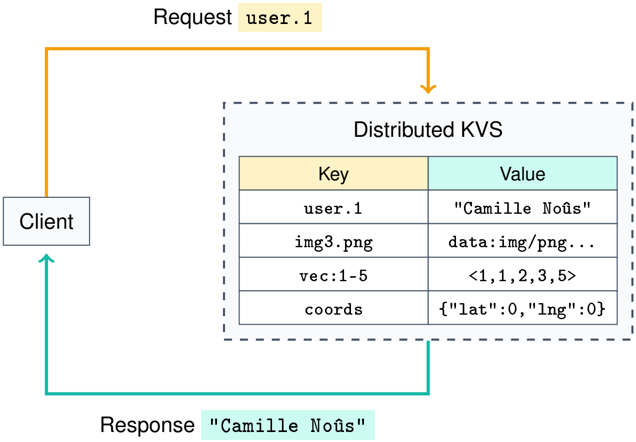
[1]LIRIS, Univ. Lyon
[2]FEMTO-ST, Univ. Franche-Comté
[3]LIP, ENS Lyon
[4]ICTEAM, UCLouvain

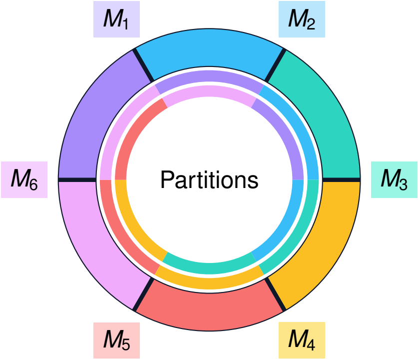
April 13, 2022 – GdT SCALE

Key-value store

Database where each value is associated to a unique key.

- Fault-tolerance
- High scalability
- High performance
- No complex queries
- No strong consistency



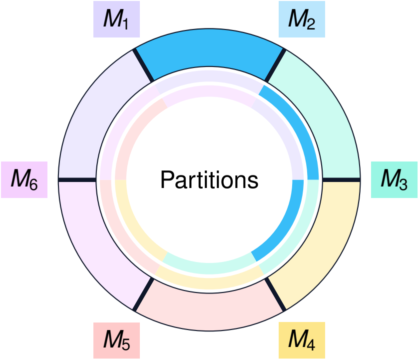


Multi-server

- Each server $M_1, M_2, \ldots$ holds a data partition.
- Partitions are replicated on different servers.
- Several servers may process a read query.

Example

Say we want to query blue partition. We may direct the operation towards M_2, M_3 or M_4.



Multi-server

- Each server $M_1, M_2, \ldots$ holds a data partition.
- Partitions are replicated on different servers.
- Several servers may process a read query.

Example

Say we want to query blue partition. We may direct the operation towards M_2, M_3 or M_4.

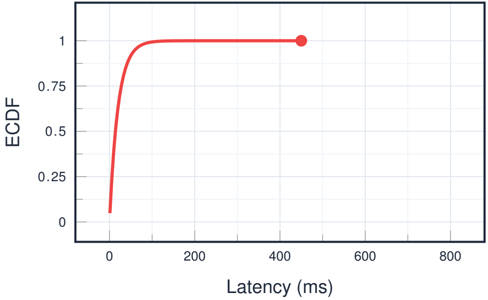
Tail latency problem

1 end-to-end request = many data items, i.e., many individual read queries.

Consequence: slowing $<$ 1% of queries may degrade the QoS for most users.

Some causes

- Background activities
- Hardware events
- Network queueing
- **Query scheduling**



Introduction

Tail latency problem

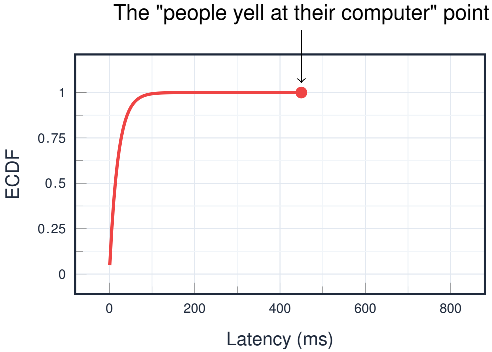
Tail latency problem

1 end-to-end request = many data items, i.e., many individual read queries.

Consequence: slowing $< 1\%$ of queries may degrade the QoS for most users.

Some causes

- Background activities
- Hardware events
- Network queueing
- **Query scheduling**



Model

Constraints

Graham's notation	Constraint
P	Homogeneous environment
$\mathcal{M}_i$	Restricted assignment
r_i	Queries arrive over time
p_i	Heterogeneous queries
$\circ$	No preemption

More constraints!

- Online model.
- Partially clairvoyant model.
- Assignment should be fast.

How to mitigate tail latency?

Idea: bound the latency of each request.

Graham's notation

$P|\mathcal{M}_i, r_i| \max w_i F_i$.

Query latency → Flow time of i: $F_i = C_i - r_i$
(C_i = completion time of i)

Bounding latency → Maximum flow $F_{\max} = \max F_i$

Other metrics → Weighted flow time $\max w_i F_i$
(e.g., slowdown/stretch)

Preemption	Class	Ref.
Non-preemptive	**NP-hard**	Immediate
Non-migratory	**NP-hard**	[Ben Mokhtar et al. 2021]
Preemptive	P	[Legrand et al. 2008]

Complexity

- Non-preemptive problem is difficult.
- Task migration necessary to make problem easier.
- Migration is hard in a real-time system.

$\mathcal{M}_i$	r_i	w_i
No	No	Yes

Highest Weight First (HWF)

1. Sort tasks in non-increasing order of w_i.
2. Put each task on the machine that completes it first.

Problem	Algorithm	Result	Ref.
$1\|\| \max w_i C_i$	HWF	Optimal	[Hall 1993]
$Q\|p_i = p\| \max w_i C_i$	HWF	**Optimal**	Not yet published
$P\|\| \max w_i C_i$	HWF	2-approx.	[Hall 1993]

$\mathcal{M}_i$	r_i	w_i
No	Yes	No

Earliest Finish Time (EFT)

When a task arrives, put it on the machine that completes it first.

Problem	Algorithm	Result	Ref.
$1\|r_i\|F_{\max}$	EFT	Optimal	[Bender et al. 1998]
$P\|r_i\|F_{\max}$	EFT	3-approx.	[Bender et al. 1998]
$R\|r_i\|F_{\max}$	LP + Rounding	$O(\log n)$-approx.	[Bansal et al. 2015]

Scheduling

Relaxed variants

$\mathcal{M}_i$	r_i	w_i
No	Yes	Yes

Problem	Algorithm	Result	Ref.
$1 \vert r_i \vert \max w_i F_i$	Any	**$\mathbf{CR} \geq \boldsymbol{\Delta} + 1$**	Not yet published
$R \vert r_i, pmtn \vert \max w_i F_i$	LP	Optimal	[Legrand et al. 2008]

CR = competitive ratio
$\Delta = \max p_i / \min p_i$

Scheduling

Relaxed variants

$\mathcal{M}_i$	r_i	w_i
No	Yes	Yes

Problem	Algorithm	Result	Ref.
$1 \| r_i \| \max w_i F_i$	Any	**CR $\geq \Delta + 1$**	Not yet published
$R \| r_i, pmtn \| \max w_i F_i$	LP	Optimal	[Legrand et al. 2008]

CR = competitive ratio
$\Delta = \max p_i / \min p_i$

Lower bound

- Let $\mathcal{I}$ be any instance of the problem.
- Let $\mathcal{S}_p^*(\mathcal{I})$ be an optimal preemptive solution.
- Then $\mathcal{S}_p^*(\mathcal{I}) \leq \mathcal{S}(\mathcal{I})$ for any non-preemptive solution $\mathcal{S}(\mathcal{I})$.

Replica selection	**Step 1**: server assignment
EFT	Select server completing the earliest
EFT-S	Same as EFT, but large queries done by specialized servers
Héron	Ref. [Jaiman et al. 2018]
LOR	Ref. [Suresh et al. 2015]
Random	Select server randomly
Local scheduling	**Step 2**: processing order on servers
FIFO	Process queries by order of arrival
MWF	When idle, process query with highest weighted flow time

Simulations

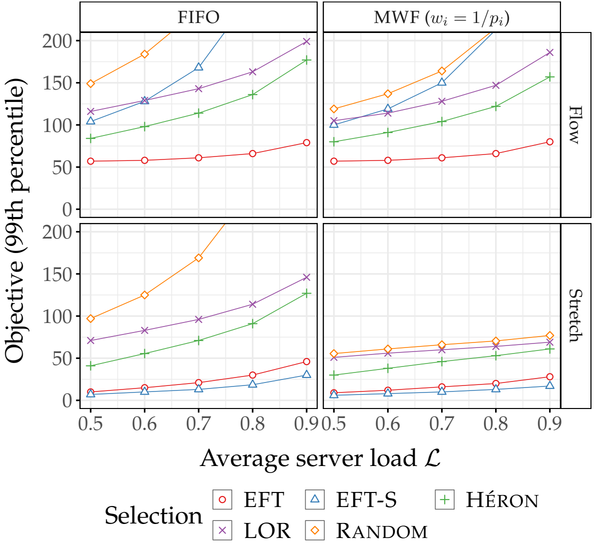
Results

Assumptions

- Stable network, no rare events
- Poisson process, heterogeneous sizes (small++), uniform popularity
- No outdated information

Experiment

- Each scenario runs for 2 minutes
- Average load vs. 99th quantile of flow/stretch

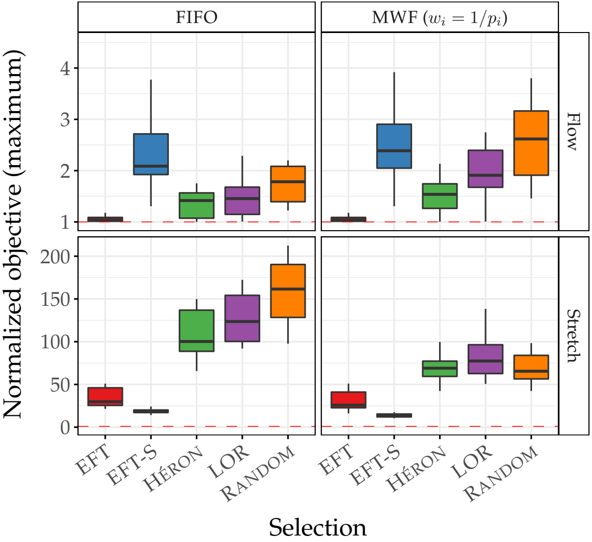


Simulations

Results

Experiment

- 1 boxplot = 10 scenarios
- 1 scenario = 1200 tasks/15 servers
- For each scenario
    1. Solve the preemptive instance with LP from [Legrand et al. 2008]
    2. **Normalize** the objective obtained from simulation
- Red line = lower bound



Simulations

Discussion

Replica selection

EFT close to lower bound. . . but hard to implement.

Local scheduling

Local policy may have positive effect on 99th quantile.

Metrics

Stretch should not be neglected to avoid delaying small queries.

Conclusion

Takeaways

- Scheduling model for key-value stores.
- Difficulty of general problem and relaxed variants.
- Perfect information allow to attain lower bound for some non-trivial inputs.

Some perspectives

- Can we compute an even tighter lower bound?
- How would a degraded version of EFT behave?
- Introduce popularity biases.
- Extend to multiget operations.