The 23rd IEEE International Symposium on Network Computing and Applications

NCA 2025

Camila Murad Veille, Lamine Amour, Scott Fowler and Sami Souihi

QoE-Driven Optimization of ZFS for Performance-Aware File Sharing Platforms

Lamine Amour

ESME Engineering School – Paris - FRANCE

lamine.amour@esme.fr

Summary

- Introduction and motivation
- System architecture
- QoE generic model
- Experimentations
- Results
- Conclusion and future works

Introduction

File systems

File systems are critical for organizing and accessing data on storage media.

System	Snapshots	Compression	Deduplication	RAID-Z / Parity
Btrfs	Yes	Yes	Yes	No
exFAT	No	No	No	No
APFS	Yes	Yes	No	No
NTFS	No	No	No	No
ZFS	Yes	Native	Yes	Yes (RAID-Z)

Early systems like ext2 and FAT3 : Limited environments, lacked advanced features,

Modern file systems like ZFS and Btrfs offer

- enhanced scalability, performance, and resilience.
- provide innovations such as snapshot management, compression, and deduplication, thereby improving data integrity and reliability

Context

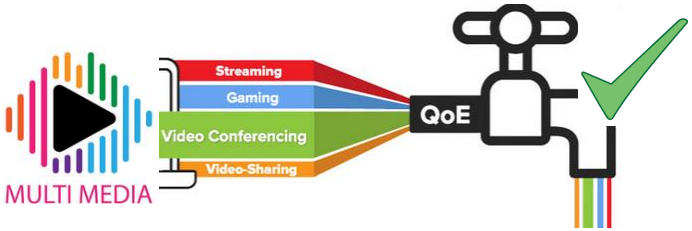
❑ QoE and File Systems

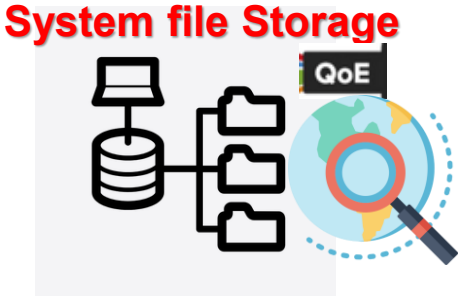
In [7], the authors propose a situational QoE model for cloud storage services based on four user studies: online survey (349) a Dropbox survey (49), a mobile study (3), and a lab study (52).

↳ Not easy to assess real QoE in the file storage systems context

In [8], authors evaluates QoE in Personal Cloud Storage and File Synchronization (e.g., Google Drive) through lab tests with 52 participants.

↳ Users tolerate brief delays for small files but report frustration with larger transfers under constrained conditions.





- Traditional approaches to Quality of Experience (QoE) optimization in file storage systems often rely on rule-based heuristics or static configurations
- Prior storage optimization has focused on static or machine learning-based performance tuning, neglecting dynamic user satisfaction modeling.

Motivation

- Main proposal : address QoE improvements for a large, uniform, decentralized file system.
- Since QoE depends on many parameters (as is explained in the paper), using machine learning is a good approach to tackling this problem.
- Propose a reinforcement learning approach and define a Q-learning method
- QoE driven Reinforcement-Learning feedback loop on top of ZFS (Zettabyte File Storage system) and wrapping it in a four module micro-service

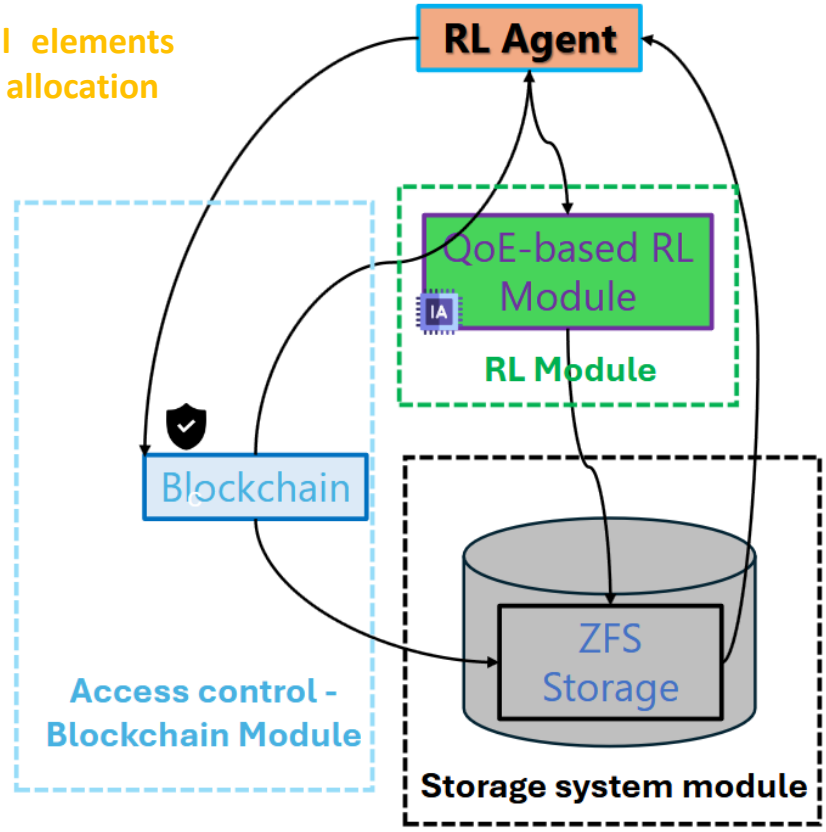
System architecture

RL Agent learns automatically to regulate critical elements such as latency, throughput, and memory allocation

QoE-based RL Module Combine QoE and RL to enable dynamic, data-driven decision-making

Blockchain This module implements blockchain-based trust mechanisms using Hyperledger Indy and Decentralized Identifiers (DID).

ZFS Storage Storage System Module for persistent data handling and the API Gateway



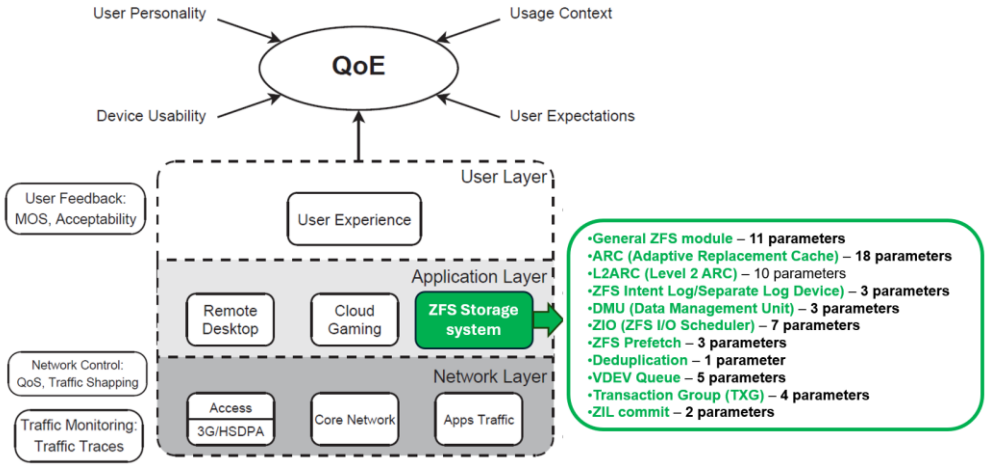
QoE Generic model

- Propose a generic QoE architectural model in the context of **System file Storage**

- *Model QoE as a function of 67 ZFS influence factors (IFs), x_i (i = 1, . . . , 67), defined in the OpenZFS (General model - Equation 2):*

$$\text{QoE} = f(x_1, x_2, \ldots, x_{67}) \qquad (2)$$

where each internal factor is denoted as x_i, with $i = 1, \ldots, 67$.



- *Users cannot provide consistent or reproducible ratings of system performance*
- *File storage QoE is only perceived indirectly through system responsiveness.*
- *QoE is approximated using a set of KQIs (Key Quality Indicators) that capture measurable aspects of performance.*

$$QoE \approx f(KQI_1, KQI_2, \ldots, KQI_n), \qquad (3)$$

where the function $f(\cdot)$ maps objective KQIs into an estimated QoE.

Summary

$$\text{QoE} = f(\text{KQI}) = f(g(x_i \mid i \in \mathcal{S})) \qquad (4)$$

where $\mathcal{S} \subseteq \{1, 2, \ldots, 67\}$ is the set of indices corresponding to the most influential parameters, $g(\cdot)$ represents the mapping from ZFS internal factors to KQI, and $f(\cdot)$ models the user perception based on the selected indicators.

- *Explored QKIs = {*read/write speed, average CPU usage rate, ARC cache usage, disk I/O latency*}*

Experimentation

Implementation

docker Compose to orchestrate multi-service deployments (e.g., API, database, monitoring)

Prometheus to collect and store granular performance metrics (e.g., read speed, CPU usage rate, latency, packet loss rates)

Grafana to provide dynamic dashboards to visualize trends, correlate anomalies, and guide optimization efforts.

Objective :

Study the impact of ZFS parameters tuning on the considered KQIs (e.g., read/write speed, average CPU usage rate, ARC cache usage, disk I/O latency, etc.)

Experimentation :

This script captures more than 100 parameters (system, network, ZFS, etc.) periodically (once per second) for one hour.

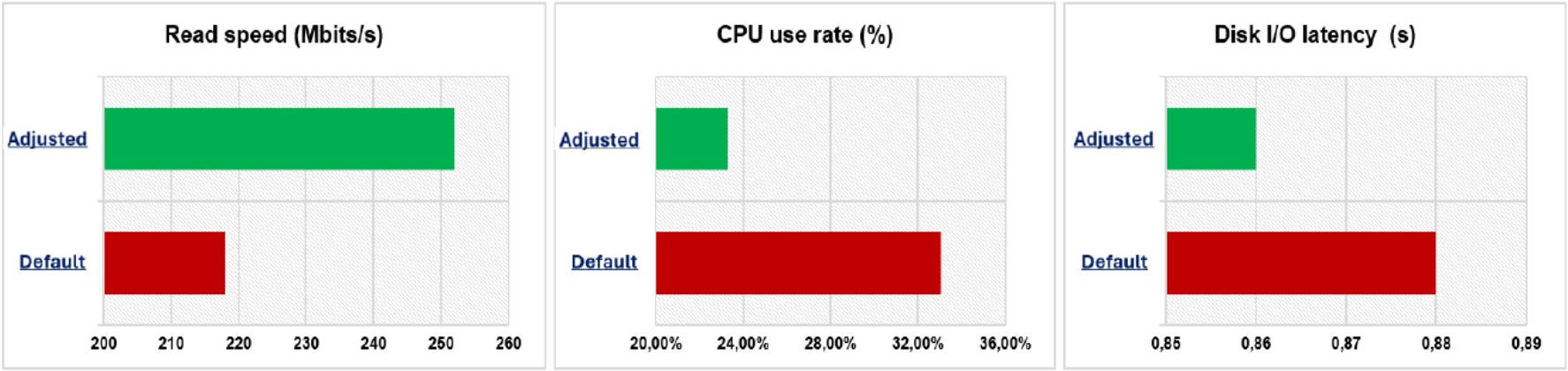
Scenario 1: Assess whether tuning ZFS parameters improves the performance of three KQIs named ”read speed”, ”CPU efficiency”, and ”disk I/O latency performance”

Scenario 2 : Benchmarks to to identify the best QoE IFs for ZFS of three KQIs : ”write speed”, ”ARC cache usage”, and ”disk I/O latency”

Scenario 3 : Compare RL algorithms to detect best ZFS QoE IFs.

Results

Scenario 1: Assess whether tuning ZFS parameters improves the performance of three KQIs named ”read speed”, ”CPU efficiency”, and ”disk I/O latency performance”



Insights :

Targeted tuning of ZFS parameters can yield substantial performance gains for sequential read operations, particularly in scenarios where throughput and resource efficiency are prioritized over ultra-low latency.

Results

Scenario 2: Benchmarks to to identify the best QoE IFs for ZFS of three KQIs : ”write speed”, ”ARC cache usage”, and ”disk I/O latency”

The test environment used in this part is as follows:

- An Ubuntu system with Docker installed.
- The ZFS file system configured with different combinations (QoE IFs)
- Tests performed on SSD and HDD disks.

QoE IFs					KQI		
Compression	Sync	Primary Cache	Secondary Cache	Disk	Write Speed (MB/s)	ARC Hits	Disk I/O (ms)
off	disabled	all	all	SSD	415	3304855	0.00
off	disabled	metadata	none	SSD	546	3272113	0.00
off	standard	all	all	HDD	497	3266636	0.00
off	standard	all	none	HDD	556	3288433	1.00
lz4	disabled	all	all	SSD	515	3250300	0.00
lz4	disabled	metadata	none	HDD	464	3277549	0.00
lz4	standard	all	all	SSD	333	3239281	0.00
lz4	standard	all	none	SSD	484	3293896	1.00
lz4	standard	metadata	none	HDD	353	3244714	0.00
lz4	standard	metadata	all	HDD	458	3310340	0.00
gzip-9	disabled	all	all	HDD	539	3299377	34.00
gzip-9	disabled	metadata	all	SSD	461	3261219	0.00
gzip-9	standard	all	none	HDD	572	3255786	1.00
gzip-9	standard	all	all	SSD	538	3315733	0.00
gzip-9	standard	metadata	all	SSD	457	3282930	0.00

Insights :

- gzip-9 compression offers an excellent compromise between write performance and ARC cache efficiency.
- Standard synchronization mode provides more stability in metrics, especially for I/O latency.
- Metadata cache is particularly advantageous for frequent metadata reads.

Conclusion :

Manual ZFS QoE IFs tuning requires domain expertise and lacks adaptability to dynamic workloads.

Exploration Strategies in Reinforcement Learning

Results

Scenario 3 : Compare RL algorithms to detect best ZFS QoE IFs.

Designed a **RL Agent** to optimize ZFS performance by dynamically selecting configuration parameters (e.g., compression, cache, disk) to maximize the selected KQI (write speed in MB/s).

Full Exploration (FE), where the agent selects configurations randomly

Constrained Q-Learning (CQL), combines exploration and exploitation, allowing the RL agent to progressively adapt its decisions through experience.

Deep Q-Learning (DQL), emphasizes deeper exploration in the initial phases, followed by exploitation once system conditions stabilize

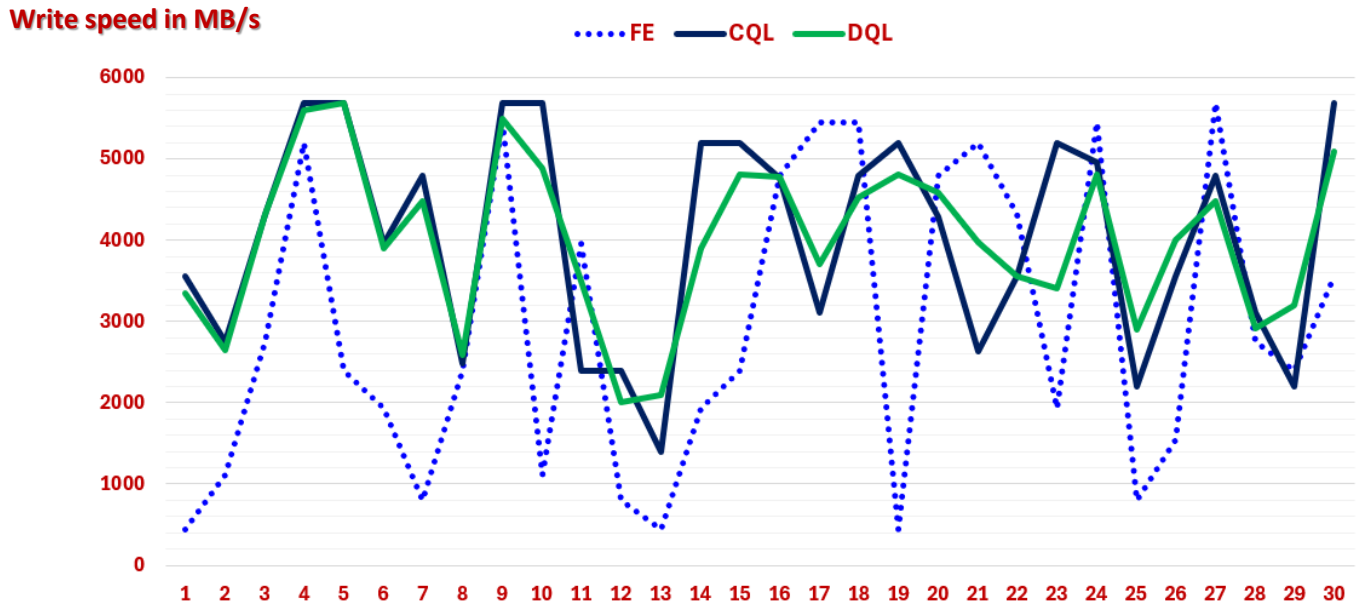
Main difference between the two RL agents, CQL and DQL, is that when applying the actions (in RL) :

- **CQL** modifies just 5 parameters of ZFS which are the QoE IFs (scenario 2),
- **QDL** modifies 27 parameters from three categories (9 parameters per category) of ZFS parameters

”Performance CPU and Memory”, ”ZFS Performance” and ”Network, System, and General Performance”

ML deployment

Scenario 3 : Compare RL algorithms to detect best ZFS QoE IFs.



- FE method presents lower and highly variable performance.
- CQL and DQL methods achieve significantly higher average write speeds
- No statistically significant difference between the two approaches Nevertheless, both methods provide stable and reliable storage performance (DQL showing a marginal higher throughput).

Conclusion

- Add a QoE driven Reinforcement-Learning feedback loop on top of ZFS
- Wrappe it in a four module micro-service stack that also logs access events on a permissioned blockchain.
- Study and propose a generic QoE IFs modeling system in the context of ZFS;
- Confirm the importance of tuning ZFS parameters and evaluate some KQI in the context of File storage
- RL provides a powerful solution for determining the best ZFS parameters
- As future works:
  - Investigate new policy based RL methods such as Actor-Critic (AC), A2C (Advantage Actor-Critic), A3C (Asynchronous Advantage Actor-Critic), and REINFORCE (Monte Carlo Policy Gradient).
  - to enhance the selection of QoE-influencing factors (QoE-IFs), either through standard ML techniques (e.g., Random Forests) or heuristics such as [2]

Camila Murad Veille, Lamine Amour, Scott Fowler and Sami Souihi

Thanks
For your attention

Lamine Amour

ESME Engineering School – Paris - FRANCE

lamine.amour@esme.fr