FABME: File-level placement policy with control admission, burst prefetching, and multi-criteria eviction for HPC multi-tier storage

Hocine Mahni, Stéphane Rubini, Sebastien Gougeaud, Philippe Deniel, Jalil Boukhobza

FABME: File-Level Placement Policy with Control Admission, Burst Prefetching, and Multi-Criteria Eviction for HPC Multi-Tier Storage

Hocine Mahni[†], Stéphane Rubini[‡], Sebastien Gougeaud[*], Philippe Deniel[*], Jalil Boukhobza[†]

[†]Lab-STICC, CNRS UMR 6285 , ENSTA, Institut Polytechnique de Paris, 29806 Brest, France,
[‡]Univ. Brest, Lab-STICC, CNRS, UMR 6285, Brest, France, *CEA, Bruyères-le-Châtel, France

1 – HPC Data Placement on Heterogeneous and Multilevel Storage System

- The global data volume will reach 394 zettabytes in 2028 [6]
- Exascale computing may widen the gap between computation, main memory and storage
- Exploiting multi-tier and heterogeneous storage systems is a key to reach trade-off between performance, cost, and capacity (Table 1, Fig 1)
- Data placement and migration between tiers is automated by hierarchical storage management (HSM)
- HSM tool (Robinhood [4]) manage data at file granularity, so block-cache algorithms cannot be used
- MC-ARC[5] **file-level** policy outperforms block caches strategies

Table 1: Comparison of the characteristics of leading memory and storage technologies [2]

Device	Read latency	Write latency	Write endurance	Cell size $(F)^2$	Cost
STT-RAM	2–35 ns	3–50 ns	$> 10^{15}$	6–50	Highest
Flash	15–35 µs	200–500 µs	10^4–10^5	4–6	\$0.5–2 / GB
Disk	3–5 ms	3–5 ms	$> 10^{15}$	N/A	\$0.06–0.3 / GB

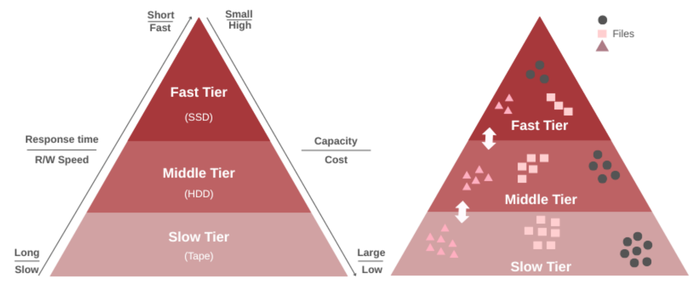

Fig. 1: Hierarchical Storage Systems (left), with HSM at the file granularity (right)

Research problem: How to design an extension to MC-ARC that integrates admission and prefetching mechanisms to optimize multi-tier storage management while preserving fairness among users?

2 – Background

MC-ARC[5] designed to decide which **files** should reside in the high-performance tier of a multi-level system. **MC-ARC** evaluates each **file** according to three complementary scores:

A1 **Usage**: to prioritize files with a large proportion of blocks accessed recently and frequently

A2 **Remaining lifetime** : to prefer files predicted to stay relevant longer, so that long-lived files remain on the high-performance tier

A3 **User fairness** : to penalize users who already consume a disproportionate share of the SSD, achieving fair resource sharing

MC-ARC can use either **WSM** or **TOPSIS** to merge the three scores

3 – Motivations

Using **MC-ARC**[1] simulator, replaying the YOMBO[2] trace with (SSD size = 0.1–5% of workload) shows that:
M1 42% of files occupy 98% of SSD yet causing 93.5% of evictions → cache pollution
M2 3.9 × SSD shuffled, taking 90% of execution time → large-file thrashing
M3 Bursty and periodic file profiles detected (see Fig. 2)

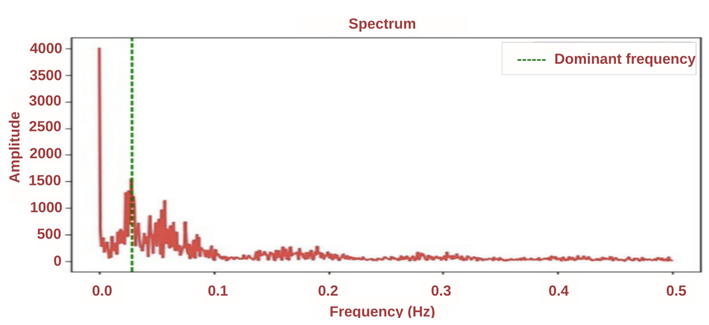

Fig. 2: Period detection on the YOMBO workload using FFT and Z-score

4a – Challenges & Contributions

Challenges: Summarize the motivations(M1, M2, M3)

C1 Cache pollution caused by very large or rarely used files

C2 Alternating large files overflows SSD tier, triggering thrashing

C3 Files characterized by short, periodic I/O bursts tend to persist in the upper-tier cache well beyond the decay of their temporal locality, resulting in sub-optimal utilization of high-performance capacity

Contributions:

❶ **Admission control**: on each miss, the policy decides to admit the file to the SSD or confine it to HDD, according to its admission rules → C1 C2

❷ **Burst prefetch**: a lightweight detector anticipates bursts and prefetches the file just before the burst starts → C3

❸ **MC-ARC updated**[5]: To manage evictions from the high-performance tier, we enhance MC-ARC to delay the eviction of files with bursty access patterns until their bursts are completed → C3

4b – Admission control

Challenges (C1,C2) : MC-ARC follows an **on single accesses** admission : every file that is accessed is promoted to the SSD tier. This maximises the hit ratio for hot data, but trigger excessive file transfers, raising I/O overhead

Proposed approach: Before any promotion (❶ in Fig.4) we evaluate three factors: F1 : **file size**, F2 : **access density**, F3 : **benefit/cost ratio**,
A file is admitted to the SSD only when these criteria indicate a net gain; otherwise it remains on HDD

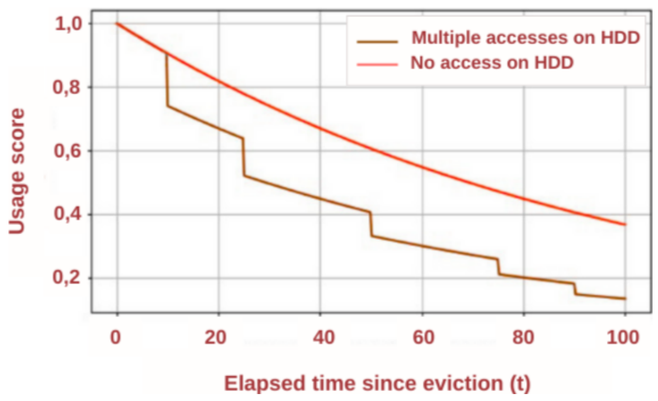

Fig. 3: Evolution of the usage score of confined files

$$f(t,a) = \text{usage}\, e^{-\varepsilon t}\, e^{-a\,\alpha t}, \qquad \varepsilon : \text{time-decay},\ \alpha_i = \tfrac{1}{\text{size}(F_i)},\ a \in \{0,1\}.$$

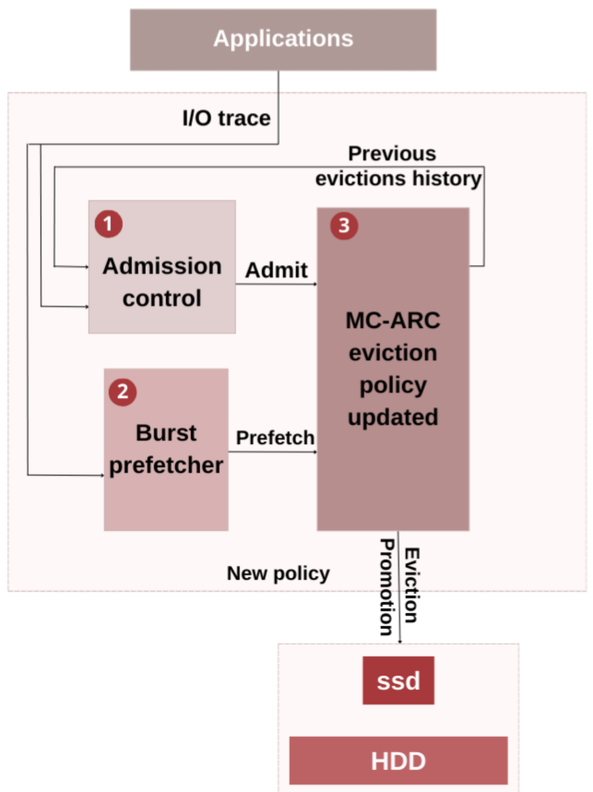

Fig. 4: Proposed solution architecture

4c – Burst prefetch & eviction

Challenge(C3): Temporal I/O bursts either miss the SSD tier's cache window or persist beyond their locality, leading to underutilization and pollution of high-performance storage

Proposed approach: A lightweight predictor (Fig.5) analyzes the I/O stream to forecast burst timings, calls AdmissionControl for just-in-time prefetching, and schedules eviction immediately upon burst completion(❷, ❸ in Fig. 4)

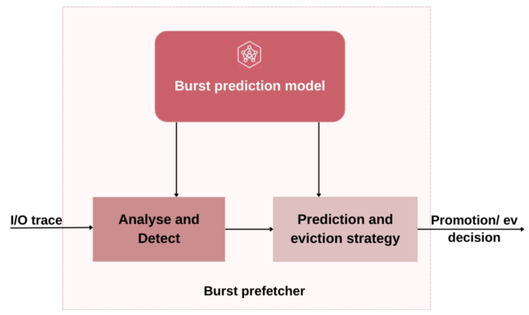

Fig. 5: Burst prefetcher architecture

5 – Perspectives

P1 **Strategy integration** : we will integrate the three modules—**admission**, **burst-aware prefetching**, and **eviction**—into the open-source simulator already used for MC-ARC evaluation

P2 **Comparative evaluation** : our proposal will be benchmarked against state-of-the-art tiering policies:[3] Amazon S3 Intelligent-Tiering, [4] Ceph Cache Tiering, and the **RejectX** admission policy used in Google's Colossus flash caches [7]

P3 **Realistic workloads** : the experimental campaign will rely on production HPC I/O traces to ensure that results reflect real-world application behaviour

The MC-ARC simulator is available at:

[1] **Amazon S3 Intelligent-Tiering**. Accessed 18 May 2025. URL: `https://docs.aws.amazon.com/AmazonS3/latest/userguide/intelligent-tiering-overview.html`.
[2] Jalil Boukhobza et al. "Emerging NVM: A Survey on Architectural Integration and Research Challenges". In: **ACM Transactions on Design Automation of Electronic Systems** 23.2 (2017), 14:1–14:32.
[3] **Ceph Cache Tiering**. Accessed 18 May 2025. URL: `https://docs.ceph.com/en/mimic/rados/operations/cache-tiering/`.
[4] Thomas Leibovici. "Taking back control of HPC file systems with Robinhood Policy Engine". In: **arXiv preprint arXiv:1505.01448** (2015).
[5] Hocine Mahni et al. "Multicriteria File-Level Placement Policy for HPC Storage". In: **Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing**. 2025, pp. 1399–1406.
[6] T. Petroc. **Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2023, with forecasts from 2024 to 2028**. Technical Report. 2024.
[7] Daniel Lin-Kit Wong et al. "Baleen:{ML} admission & prefetching for flash caches". In: **22nd USENIX Conference on File and Storage Technologies (FAST 24)**. 2024, pp. 347–371.

9th edition of the Workshop
Performance and Scalability of Storage Systems
May 23 2025 Maison des Mines et des Ponts, Paris, France