TeraCache: Efficient Caching over Fast Storage Devices

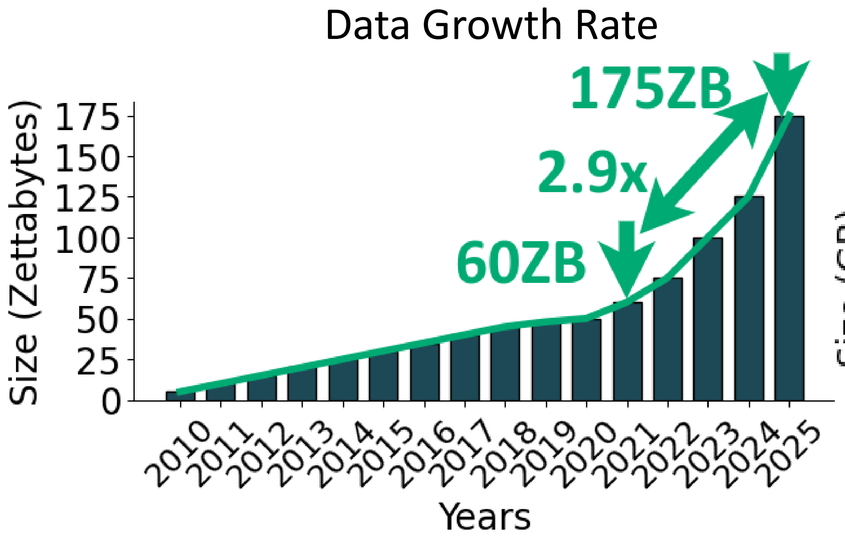
Iacovos G. Kolokasis[1,2], Anastasios Papagiannis[1,2], Foivos Zakkak[3],
Shoaib Akram[4], Christos Kozanitis[1], Polyvios Pratikakis[1,2], and Angelos Bilas[1,2]

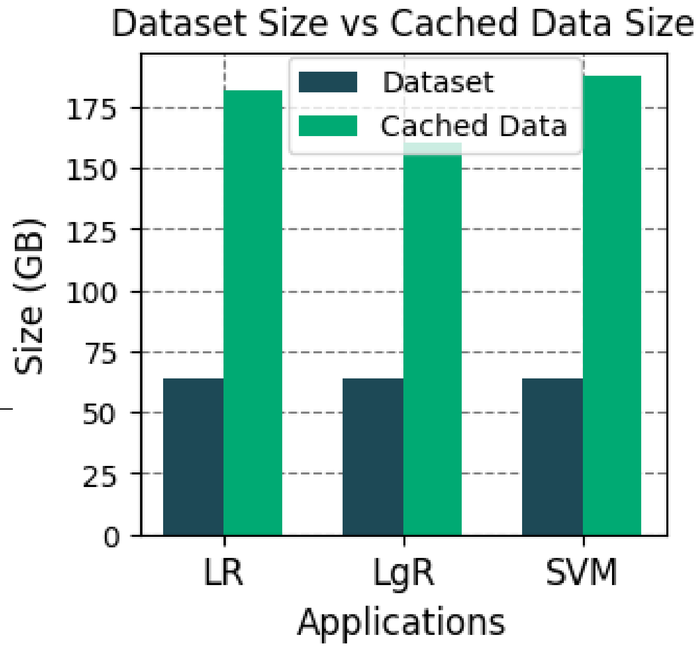
[1]Foundation for Research and Technology – Hellas GR, [2]University of Crete GR, [3]Red Hat Inc., [4]Australian National University AU

Fast Storage Devices Available but Analytic Stacks Not Ready

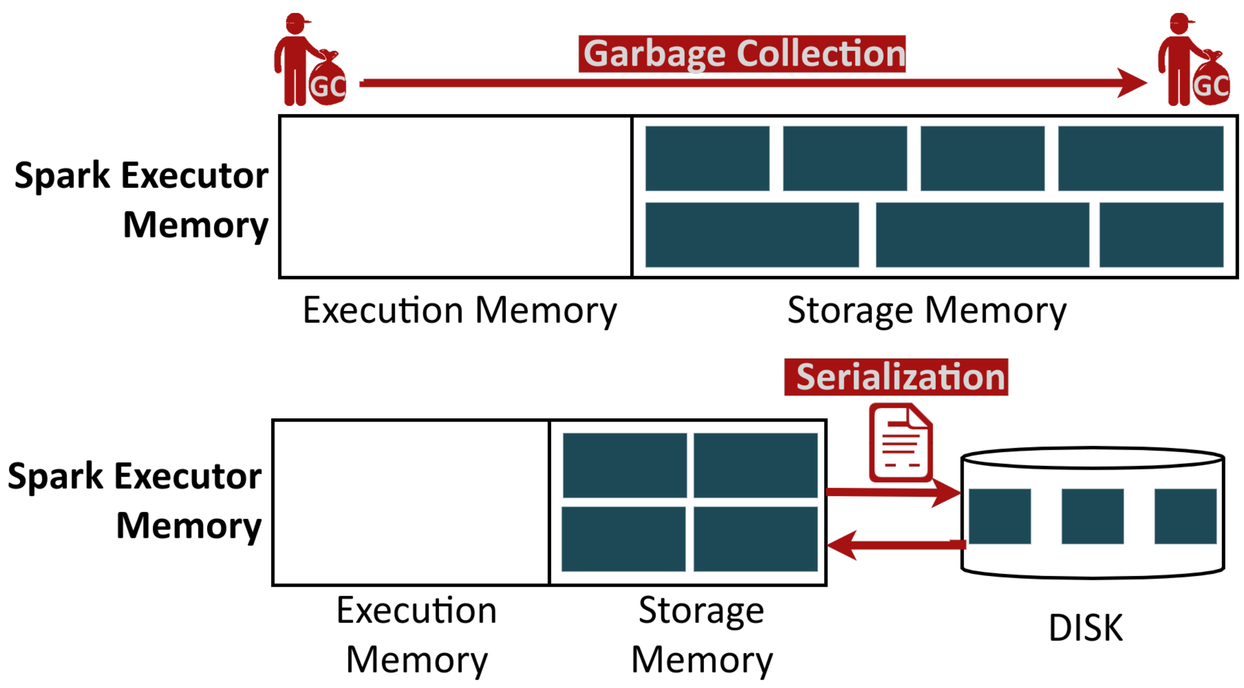
Increasing Memory Demands for Analytics

- Analytic servers use caches for avoiding recomputation (compute caches)
- Cache size is often several times the input dataset size
- DRAM scaling is limited (more $ per GB)
- Analytics resort to high capacity, fast storage devices, such as NVMe



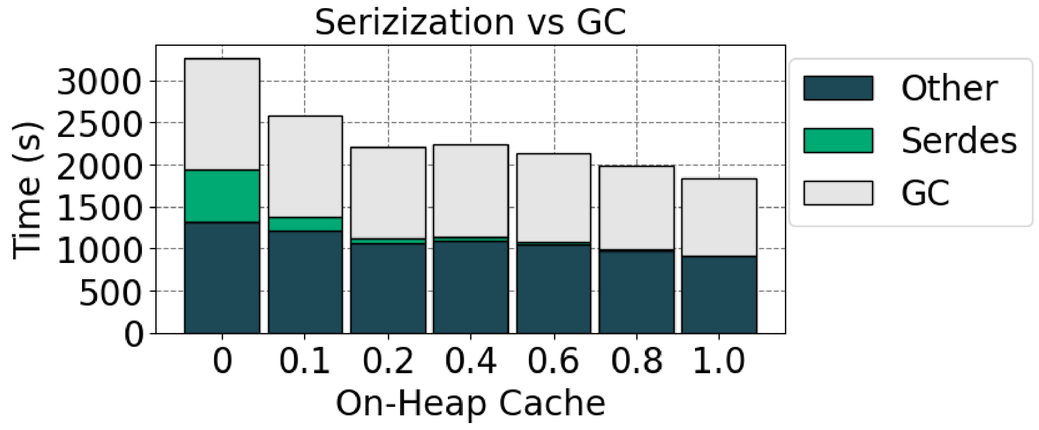


Spark Offers On-Heap and Off-Heap Caching

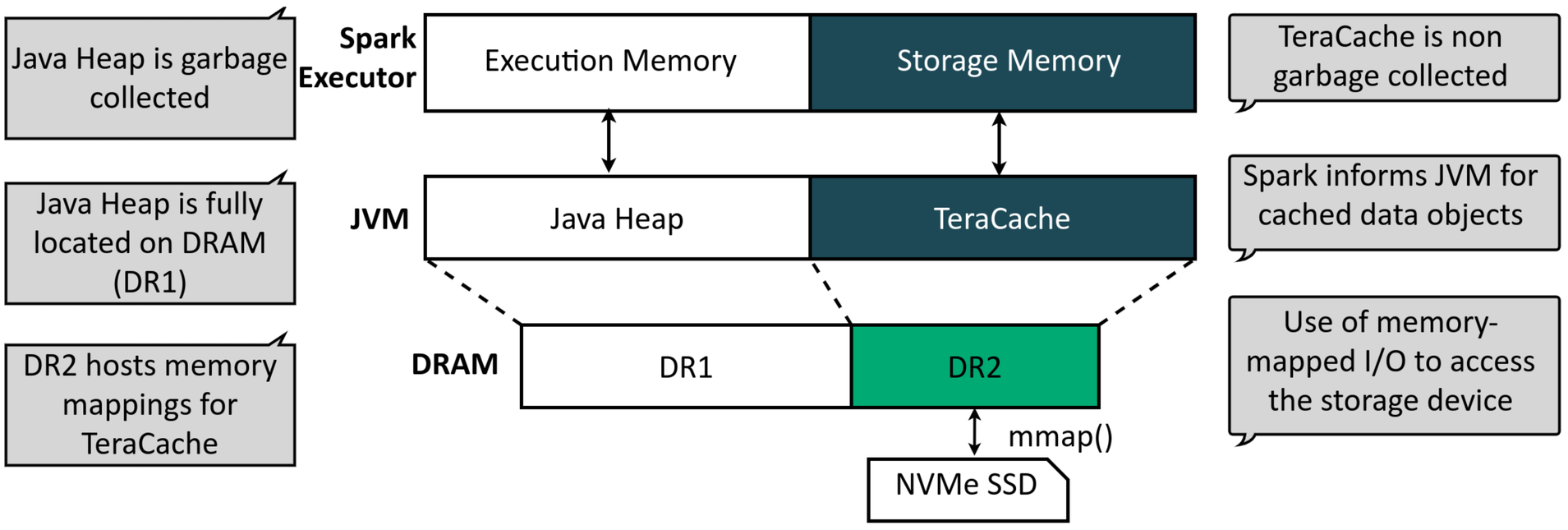


Merging On-Heap and Off-Heap Benefits

	Pros	Cons
On-heap	No Serialization	High GC Time
Off-heap	Low GC Time	High Serialization



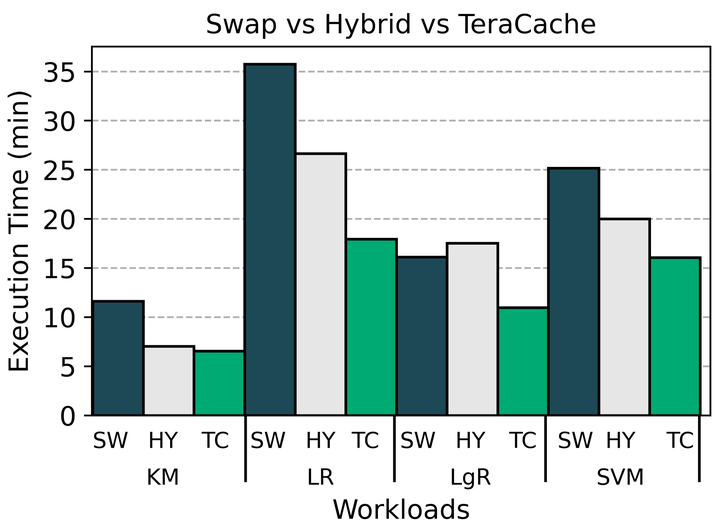
TeraCache: Best of Both Worlds!

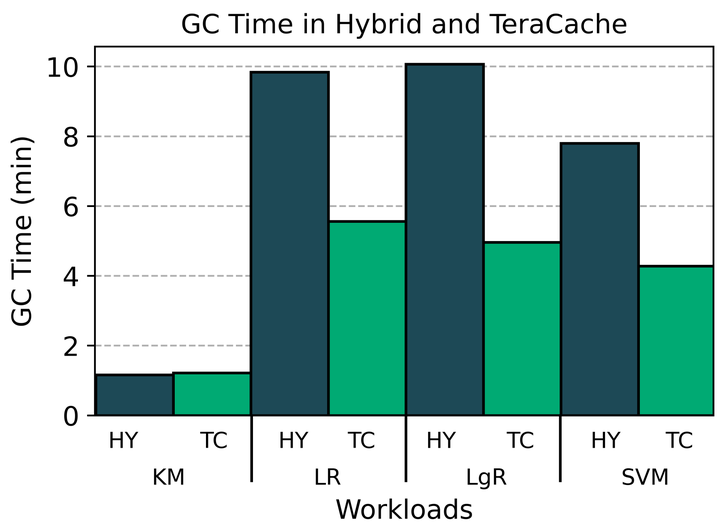


Challenges

- Interaction between Spark and JVM
- Keep classes loaded
- Keep reachable objects from TeraCache alive
- Extend post write barriers
- Extend Interpreter and JIT to support updates in TeraCache

Machine Learning Workloads





Key Takeaways

- RDD caching is critical in Spark
- GC and serialization introduce significant overhead
- TeraCache improves ML workloads performance by 25% over the state-of-the-art