Cloud-computing platforms provide services to consumers through multiple service-delivery models. Recent advances in these models have led to the emergence of serverless computing, or simply serverless, a paradigm in which software systems are composed of functions - reusable, lightweight units of code executed within isolated sandboxed environments. Major cloud-computing platforms, including Amazon Web Services (AWS), Microsoft Azure, and Google Cloud, report that a substantial proportion of their customers employ serverless solutions.

Most cloud-computing providers employ a pay-as-you-go billing model. Inefficient utilization of computing resources, particularly CPU time and working memory, which constitute the most costly resources, leads to increased overall operational costs. Moreover, the requirement for sandboxing adversely affects initialization latency and results in additional CPU and working-memory overhead. Serverless sandboxes are typically deployed on top of heavyweight virtualization stacks, further increasing working-memory consumption.

Modern cloud-computing architectures use Checkpoint/Restore techniques to freeze initialized sandboxes into a continuable form. Such techniques, in combination with cloud-native deployments, allow the virtualized environment to optimize resource consumption and share code and pre-initialized data across multiple sandboxes. However, such solutions operate at application-build time and cannot pre-initialize and share data available during application execution. Such data is processed multiple times and duplicated in each sandbox.

This dissertation presents Doss, a direct object snapshotting and sharing system that performs data c/r during application execution. Doss persists data directly, without transformations, into reusable and shareable snapshots. Direct snapshotting allows Doss to achieve near-constant data deserialization, greatly improving initialization times and reducing CPU usage. Doss architecture enables snapshot sharing across application instances, eliminating the excess memory footprint associated with data re-processing and duplication.

GraalDoss, a Doss implementation for Java, is integrated into the GraalVM ecosystem. GraalDoss is evaluated using 106 correctness and robustness tests and a novel set of cloud-native micro and macro benchmarks that exercise real-world scenarios. A comprehensive evaluation of GraalDoss shows a consistent near-constant data-deserialization overhead with serialization times comparable to state-of-the-art Java JSON and binary serialization libraries. GraalDoss eliminates the memory footprint of web API microservice caches by sharing populated cache snapshots across microservice instances, improving the overall density by 41% for 8 microservice instances and improving first-response times by 34%. In NLP applications, GraalDoss improves the pipeline execution times by six orders of magnitude by snapshotting pipeline results and subsequently loading the snapshots.