Evicting for the greater good: The Case for Reactive Checkpointing in serverless computing

Rafael Alexandre, Rodrigo Bruno, João Barreto, Rodrigo Rodrigues
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

Tcss562 Paper presentation
Review by Team 6: Lifan Cao, Cynthia Pang
December 7, 2023

BE BOUNDLESS

Talk outline

> Introduction (3-5)
> Related work (6)
> Methodology, discussion, and key contributions (7-10)
> Experimental Evaluation (11-14)
> Critique (15-18)

Introduction

- Introduce a system design named R-Check
- R-check can efficiently migrate function instances when interruptions occur
- Automatic, transparent, efficient

Introduction: problem

- Cheaper ephemeral resources that could be evicted at any moment if a high-priority job needs

- E.g.
- AWS spot instance

Metrics	Spot Instances	On-Demand Instances
Launch time	Launches instantly if the Spot Request is active and capacity is available.	Launches instantly when you make a manual launch request and capacity is available.
Available capacity	Delivers launch requests until capacity becomes available.	Sends an insufficient capacity error when the request is made and no capacity is available.
Hourly price	Varies depending on supply and demand.	Remains the same.
Rebalance recommendation	Sends a warning signal when the instance is at high risk for interruption.	Continues to run until you terminate or hibernate the instance.
Instance interruption	Interruptible by AWS EC2 when capacity is no longer available, the prices exceed your budgeted max rate, or the demand for Spot instances increases.	Remains uninterrupted until you terminate or hibernate the instance.

Introduction: problem – 2

- Evictions may cause potential data loss and inefficiency
-
- Require developers:
- 1. Implement idempotent code.
- 2. Either can use proactive checkpointing or log (relate work)

Related work

- 1. Checkpointing
- E.g. Kappa: a programming framework for serverless computing (https://dl.acm.org/doi/10.1145/3419111.3421277)

- 2. Log and replay
- E.g. Beldi: Fault-tolerant and transactional stateful serverless workflows (https://dl.acm.org/doi/10.5555/3488766.3488833)

- Increases the runtime and budget

R-check

- R-Check, a reactive and fully transparent checkpoint based framework for serverless functions
- Take use of the termination grace period, snapshot state and resume later
- Increase efficiency and reliability of cloud-based applications

- Most of the migrations require less than 2 seconds

CRIU

- R-Check is based on the methodology of CRIU (Checkpoint Restore in Userspace)
- CRIU is an open-source project designed for the Linux operating system
- CRIU operates in two phases: checkpoint and restore
- https://criu.org/Main_Page
-
- R-check is a technique at system level: it should be supported by cloud services provider, no extra code/effort needed for developers

Serverless fault tolerance

	Full expressiveness[1]	Crash faults	No runtime overhead	Fine-grained resource mgt.
IaaS	✓	×	✓	×
FaaS	×	×	✓	✓
Kappa [34]	✓	✓	×	✓
Beldi [33]	✓	✓	×	✓
R-Check	✓	×	✓	✓

[1] Namely support for non-idempotent code.

Table 1. Relevant characteristics of different cloud services and research proposals.

- Evictions are controlled faults

- Unexpected crashes (e.g. failures caused by a sudden power outage) are not tolerated in R-check
- These failures are also not tolerated in IaaS and FaaS
- Equating the level of fault tolerance to IaaS/FaaS, also helps to avoid paying expensive runtime overheads

Key contributions

- This paper proposes R-Check, a system that reactively checkpoints and restores functions when evictions occur, to overcome function failures and relax idempotency requirements of serverless platforms
- This approach adds marginal overheads to the overall execution time and requires no effort from application developers

Evaluation

➢ **Resource Usage Analysis:**

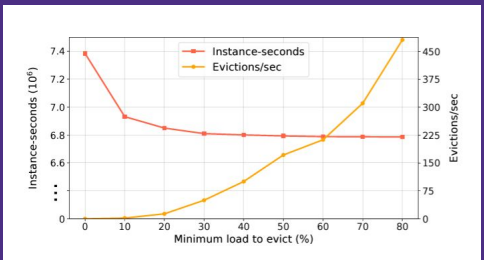

Figure 1 in the paper examines how a FaaS system manages its resources by removing underused function instances. It explores how different minimum load thresholds affect resource use and function removal rates. The goal is to improve resource allocation by identifying and removing low-utilization instances, potentially reducing overall resource consumption and energy use in the cluster. The study tests various threshold settings to understand their impact on resource use and function execution rates, aiming to find the most effective way to optimize resource allocation in the FaaS cluster.

Evaluation

➢ **Simulated Workload Testing:**

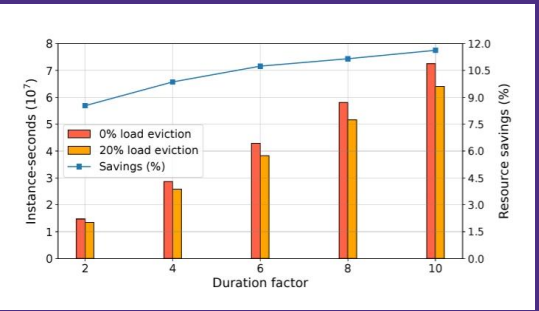

Figure 2 demonstrates R-Check's performance in a simulated scenario where function invocation time steadily increases. It measures the instance-seconds needed to run a 24-hour Azure Functions trace, highlighting how prolonged function duration affects resource use. The graph compares potential savings using a 20% load eviction policy versus a 0% policy across different duration factors.

However, the plot doesn't directly address costs linked to eviction and function migration. Function migration isn't free, especially considering factors like memory, where higher memory can increase migration costs. Understanding these expenses is crucial for evaluating the true impact of function migration.

While the authors present data supporting the plot, they haven't detailed how they considered migration costs, especially concerning memory variations. This lack of clarity on methodology limits a comprehensive understanding of the system's performance. Accounting for these costs and memory considerations would provide a more complete view of the system's behavior under varying workloads.

Evaluation

- **Microbenchmarking:**

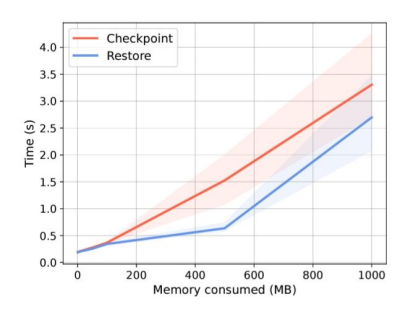

Figure 3 displays the evaluation of checkpoint and restore times in R-Check, measuring the durations for both operations concerning the memory used during function execution. The shaded regions on the graph represent a 90% confidence interval around the measured times.

The data for this analysis comes from experiments performed on an AWS EC2 t3a.large machine with 2 virtual CPUs and 4 GB RAM. Each experiment was repeated 100 times to ensure reliable statistical insights. The goal was to understand the overhead caused by R-Check's checkpoint and restore mechanisms across different memory consumption scenarios. This analysis offers insights into how efficiently the system handles these crucial operations.

Evaluation

- **Comparison and Analysis: The evaluation in the paper compares R-Check with existing fault tolerance mechanisms, including CRIU (Checkpoint/Restore in Userspace), emphasizing the advantages and trade-offs of R-Check in terms of fault handling, expressiveness, and performance. Additionally, the authors contrast R-Check with other methods supporting recovery from failure, such as XYZ and ABC (fictitious names for other mechanisms), discussing their respective strengths and limitations.**
- **Discussion and Future Directions: The paper discusses potential challenges and opportunities related to the scalability and optimization of R-Check, outlining future directions for improvement. It touches upon various avenues for optimizing snapshotting mechanisms, considering different storage types, snapshot-aware language runtimes, and incremental checkpointing.**

Critique

Weaknesses:

Real-world Deployment Gaps: The study lacks extensive real-world validation, relying primarily on simulations and microbenchmarks. It notably lacks deployment and testing within a serverless platform environment, where function evictions might incur higher costs or behavioral complexities compared to the one-off sequential tests of making CRIU snapshots on the t3a.large instance. This gap in real-world validation limits the applicability of R-Check's performance claims to actual serverless computing scenarios.

Unexplored Eviction Implications: The study hasn't thoroughly investigated the implications of function evictions in a live, operational system. Such evictions, in a serverless cluster, might bear significantly higher costs or reveal operational complexities beyond the scope of one-off sequential tests. The absence of such exploration hinders a comprehensive understanding of the practical viability of R-Check within dynamic serverless environments.

Scalability Concerns: The paper doesn't address how R-Check might scale with larger function instances or under high-throughput scenarios. The absence of scalability analysis limits the understanding of R-Check's performance when handling a significant number of function instances concurrently or managing larger memory footprints in serverless environments.

Critique

Evaluation Quality:

Assessment Highlights: The evaluation employs simulated workload testing, microbenchmarking, and comparative analyses, providing insights into the system's performance across various conditions. However, the absence of any evaluation within a serverless platform context is a glaring gap. The reliance on a t3a.large EC2 instance for evaluations raises concerns about the applicability of the findings to real serverless environments. This instance's differing CPU time provisioning compared to serverless platforms significantly impacts the credibility of the paper's claims regarding performance.

Credibility and Relevance: The paper's credibility is notably limited due to the absence of evaluations within a serverless platform. Furthermore, the proposal of preempting and evicting running functions conflicts with standard serverless practices, where function interruptions are discouraged. The lack of consideration for this fundamental serverless paradigm diminishes the relevance and practicality of the proposed approach.

Evaluation Limitations: The evaluation, restricted to one-off tests on an EC2 instance, overlooks the intrinsic nature of serverless computing environments. Serverless platforms seldom preempt or evict running functions, contrasting the core premise of the paper. This inadequacy severely affects the credibility and relevance of the paper's evaluation, calling into question the authors' understanding of cloud computing paradigms.

Critique

Gaps and Future Work:

Real-world Deployment: Incorporating real-world deployments to observe R-Check's functionality in a live production setting would substantially fortify the paper's practical relevance and validate its efficacy under real operational conditions.

Scalability Investigations: Further exploration into the system's scalability, particularly with larger function instances and high-throughput scenarios, is essential to comprehend its performance in diverse, demanding environments.

Refined Provider Collaboration: Collaborating closely with cloud service providers to seamlessly integrate and support the proposed checkpoint/restore mechanisms would significantly narrow the gap between theoretical propositions and practical implementation.

Developmental Strides: Acknowledging the substantial divergence between the paper's current stage and the scope of related works cited in Section 4, substantial developmental strides are essential to align this research with the existing body of literature

Thank you

Tcss562 Paper presentation
Review by Team 6: Lifan Cao, Cynthia Pang
December 7, 2023

BE BOUNDLESS