Faster and Cheaper Serverless Computing on Harvested Resources

Authors: Yanqi Zhang, Íñigo Goiri, Gohar Irfan Chaudhry, Rodrigo Fonseca, Sameh Elnikety, Christina Delimitrou, and Ricardo Bianchini

Presenters: Bob Schmitz, Danielle Lambion, and Viktoriya Grishkina

December 2, 2021

BE BOUNDLESS

Outline

> Introduction
> Background/Related Work
> Summary of Technology
> Key Research Contributions
> Experimental Evaluation
> Conclusions
> Paper’s Strengths / Weaknesses
> Evaluation of Paper
> Identifying Gaps
> Questions?

UNIVERSITY *of* WASHINGTON

Introduction

What's the problem?

- Serverless providers are required to manage the underlying VMs used for hosting serverless requests
- The driving factor of costs for providers is tied to resources that need to be allocated for serverless functions
  - Serverless providers must must maintain high reliability and performance while keeping cost low

Introduction

What can be done to keep cost down for serverless applications?

- Use Harvest VMs [1]
  - A proposed VM class in a published research paper by Microsoft (not currently available to the public)
  - Available for much cheaper due to relaxed guarantees of availability, similar to Spot requests
  - Can provide better performance than Spot Request and regular VMs because resources will grow or shrink based off availability on the host server
  - 30 second warning prior to eviction

[1] AMBATI, P., GOIRI, Í., FRUJERI, F., GUN, A., WANG, K., DOLAN, B., CORELL, B., PASUPULETI, S., MOSCIBRODA, T., ELNIKETY, S., AND BIANCHINI, R. Providing SLOs for Resource-Harvesting VMs in Cloud Platforms. In OSDI (2020).

Introduction

What are the challenges of using Harvest VMs for FaaS?

- Harvested resources are evictable
  - A mixture of “regular” VMs and Harvest VMs may be required for high reliability
- Managing Harvest VMs variability (i.e. hardware, heterogeneity) depends on designing an effective load balancer
  - OpenWhisk an open-source FaaS platform was used for both managing load balancing and monitoring resources

Background/Related Work

Serverless computing and FaaS

- Serverless provides the ability to upload code for applications without having to manage underlying resources
- Serverless providers must have resources at the ready whenever a function is executed
- Users only pay for resources utilized while running FaaS
- A published paper has shown that 50% of FaaS functions run for less than 1 second and 90% run less than 10 seconds on average

  https://www.microsoft.com/en-us/research/uploads/prod/2020/05/serverless-ATC20.pdf

Background/Related Work

Harvest VMs

- New proposed class of virtual machine resource
- vCPU and memory will grow or shrink based off availability on the host server

https://www.microsoft.com/en-us/research/uploads/prod/2020/09/HarvestVMs-SLOs-OSDI20.pdf

Apache OpenWhisk

- OpenWhisk is an open-source FaaS platform which allows users to monitor resources and manage load balancing
- Several works have been published on scheduling, however they assumed constant resources (unlike Harvest VMs)

Summary

Harvest VMs Setup

- A 14-day period was selected as a trial period for collection metrics (traces) from Harvest VMs and serverless workflows
- 37 harvest VM instances
- To match the minimum memory of 16GB, vCPU count was limited to 32
- The average vCPU change was 12
- The maximum vCPU size was 30
- More than 90% of the Harvest VMs run longer than a day (w/o eviction)
- The majority of invocations of FaaS executions (86%) are shorter than 1 sec., the longest one is a little less than 10 minutes

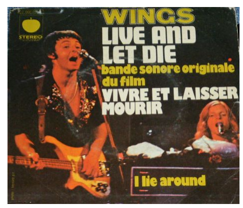
Summary

Methodology for Handling Harvest VMs Variability

> Strategy 1: No Failures

> Strategy 2: Bounded failures

> Strategy 3: Live and Let Die



Summary

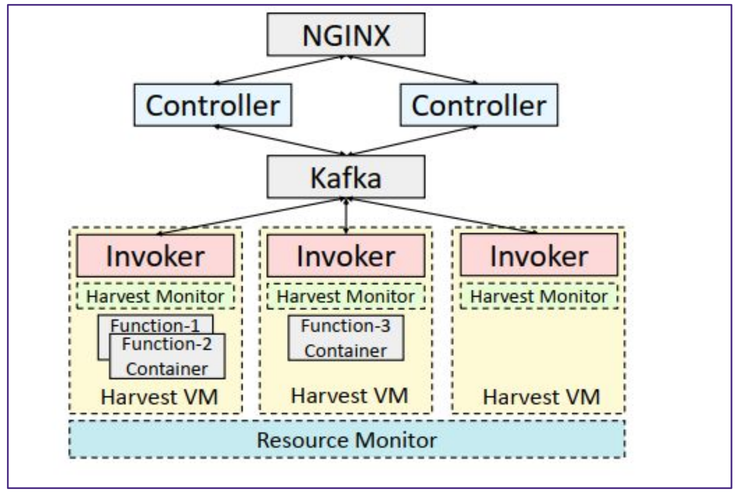
Developing/Implementing an Effective Load Balancer

- “Vanilla” OpenWhisk load balancer
- Join-the-Shortest-Queue (JSQ)
  - Monitors the compute load of each backend VM
  - Authors approximated pending compute work with where $w_c > w_m$ $$w_c \frac{cpu_{used}}{cpu_{avail}} + w_m \frac{mem_{used}}{mem_{avail}}$$
  - Distributes the work to the least utilized VM
- Min-Worker-Set (MWS)
  - Distributes to a smaller set of VMs

Summary

OpenWhisk Implementation

> Modifications to this FaaS platform are represented with a dotted line
> Invokers are deployed one per VM to manage containers
> Harvest Monitor modules are deployed to gather
  – CPUs allocated
  – cumulative CPU time
  – scheduled deallocation event
> A Resource Monitor module is used to track the resource variation in our system
> The Invoker and Controller implement the resource variation-aware MWS algorithm

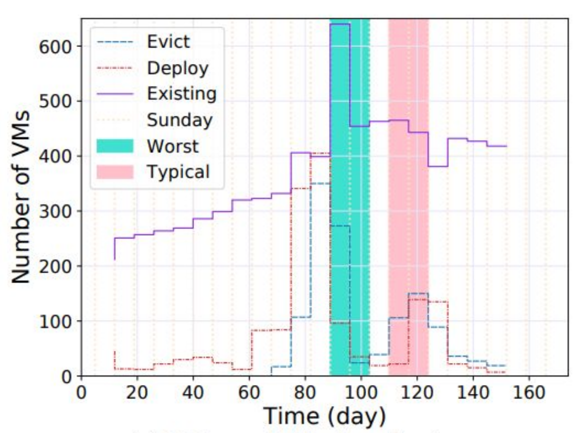


Key Contributions

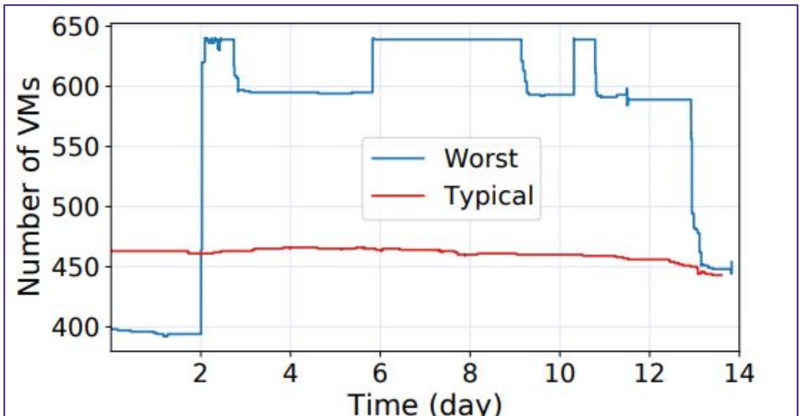
1. FaaS are much cheaper cost on Harvest VMs compared to regular VMs
   – harvested resources achieve 48% to 89% cost savings compared to regular VMs
2. Performance of FaaS is better on Harvest VMs versus regular VMs
   – harvested resources achieves 2.2× to 9.0× higher throughput compared to regular VMs due to the ability to consume more vCPU and memory when available
3. Min-Worker-Set (MWS) load balancer algorithm is shown to be effective at managing Harvest VMs variability (i.e. vCPU/memory)
   – 22.6× higher throughput compared to “vanilla” OpenWhisk load balancer due to addressing resource variability

Experimental Evaluation
Handling Evictions

- If an eviction occurs, a running function will fail
- Eviction rate = # of evictions / # of existing VMs
- The average eviction rate over 14 days is 13.1%



(a) 14-day period in the entire trace.



(b) Selected 14-day periods for simulation.

Experimental Evaluation
Handling Evictions

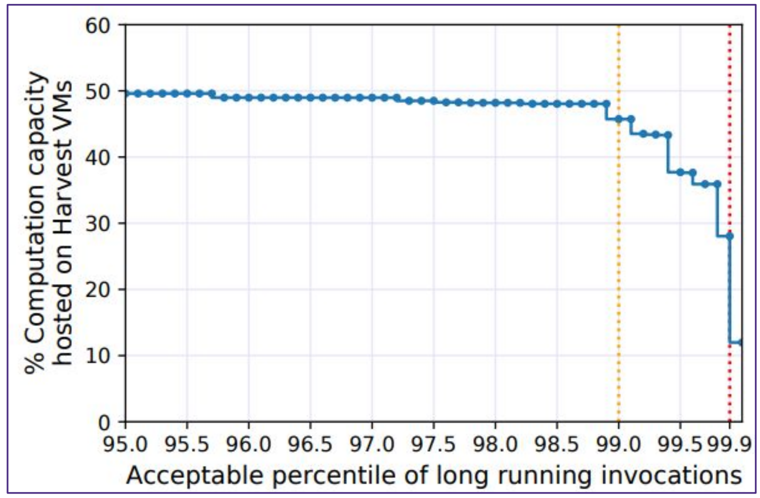
Strategy 1: No eviction failures

- Long applications (>=1 invocation >30 seconds) are on regular VMs, all else on Harvest VMs
- Least efficient provisioning strategy and high operational cost
- 94% of invocations on regular VMs are still short

Experimental Evaluation
Handling Evictions

Strategy 2: Bounded failures

- Provide an upper bound of acceptable evictions per application (e.g. 1%)
- Allocate regular VMs to applications that are in the x^{th} (e.g. 99th) percentile duration
- 94% of invocations on regular VMs are still short



Experimental Evaluation
Handling Evictions

Strategy 3: Live and let die

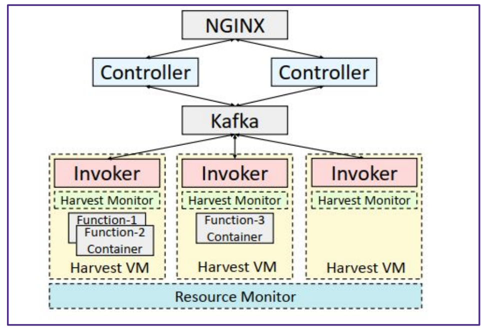
- Everything is on Harvest VMs
- Average invocation failure rate is 0.0015%
- The typical period has a failure rate of 3.68×10^{-8}
  - 7 nines of reliability (99.99999% reliable)
- Eviction is rare
  - Requires two low probability events to occur simultaneously
    - A long invocation is running
    - An eviction occurs

Experimental Setup

- OpenWhisk on Azure with Ansible
  - On real Harvest VMs and traces
- 1 Controller VM contains:
  - Core OpenWhisk components
  - NGINX
  - CouchDB
- Variable number of invokers with their own VMs
- Table 2 Python functions are used for benchmarking
- Each experiment runs for 20 minutes, unless otherwise specified

Functions	Description
Floatop	Sine, cosine & square root
Matmult	Square matrix multiplication
Linpack	Linear equation solver
Chameleon	HTML table rendering
Pyaes	AES encryption & decryption
Image processing	Flip, rotate, resize, filter & grayscale images
Video processing	Grayscale video
Image classification	MobileNet inference
Text classification	Logistic regression

Table 2: The examined serverless functions from FunctionBench [32] and their description.



Evaluation

Impact of Load Balancing

- OpenWhisk with 10 invokers
- Each hosted by Regular VM with:
  - 32 vCPUs
  - 128 GB memory
- Each invoker varies between 5-28 CPUs

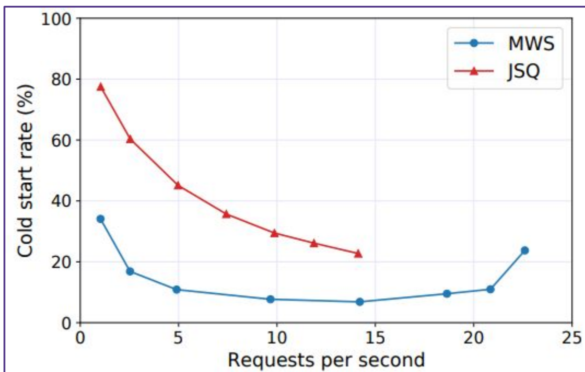


Figure 13: Cold start rate of MWS vs. JSQ.

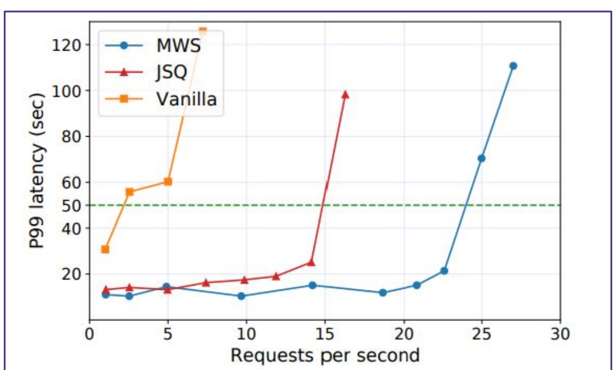


Figure 12: P99 latency across load balancing algorithms.

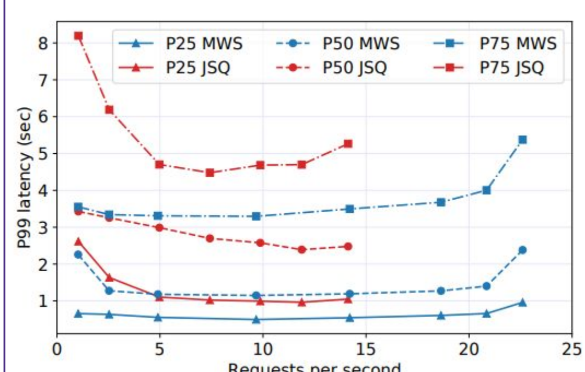
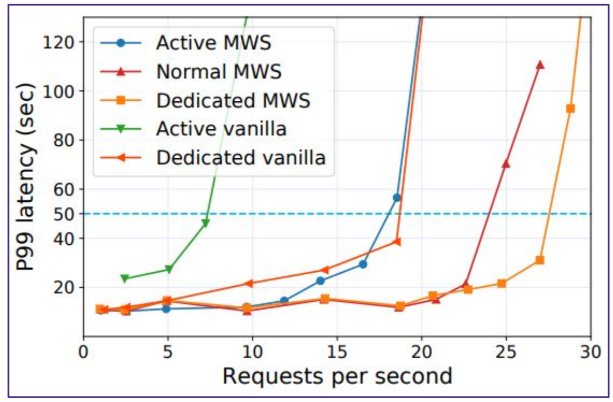


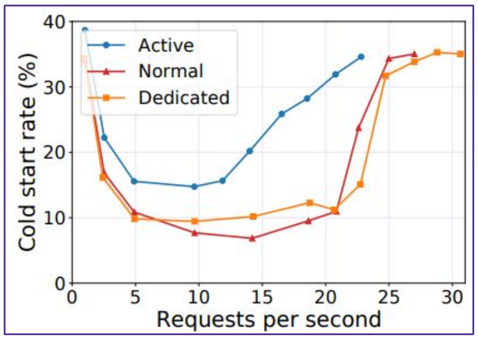
Figure 14: Low percentile latency of MWS vs. JSQ.

Evaluation

Impact of Resource Variability

- > "Active" denotes a Harvest VM cluster with significant CPU changes
- > "Normal" denotes a Harvest VM cluster with normal variation
- > "Dedicated" denotes a dedicated cluster using regular VMs





UNIVERSITY *of* WASHINGTON

Evaluation

Cost vs. Performance

Budget is compared to cost of 2 Regular VMs with:

- > 16 CPUs
- > 64 GB

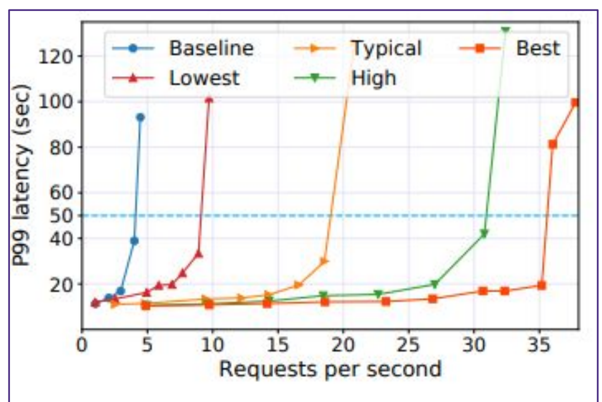
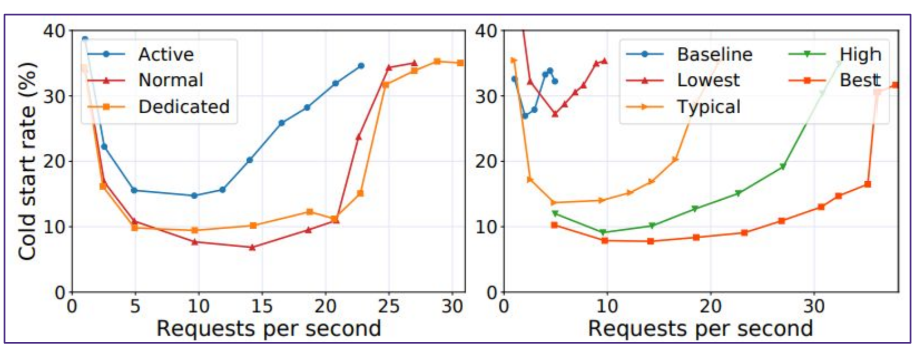


Figure 17: Regular vs Harvest VMs with same budget.



Discount	$d_{evict}(\%)$	$d_{harv}(\%)$	#VMs
Baseline (dedicated)	0	0	2
Lowest	48	48	6
Typical	70	80	12
High	80	90	18
Best	88	90	21

Table 3: Number of Harvest VMs with the same budget, based on the discount level.

Evaluation
Harvest vs. Spot VMs

- Invocation failure rate are higher on VMs with more CPUs and more often on spot VMs
- CPU sensitivity is higher on Harvest VMs and decreases as the number of CPUs increases
  - CPUs X time normalized with the cluster's idle CPUs X time
- Spot instances cost more
  - H2 offers 0.211$/hour
  - Lowest per-CPU price of Spot VM is 0.313$/hour

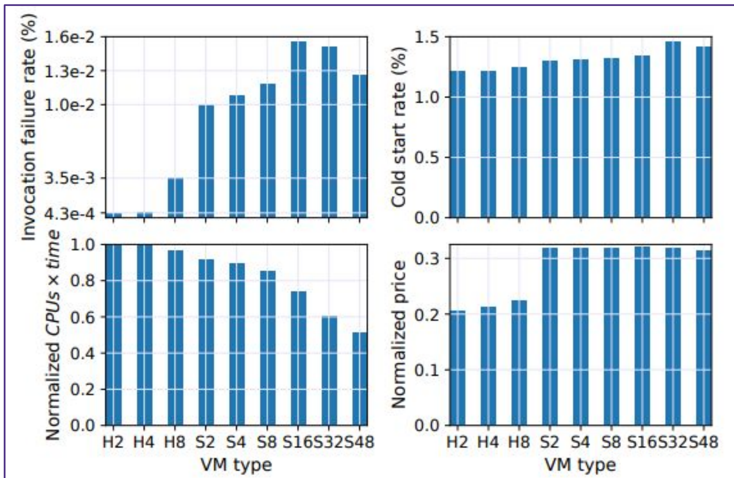


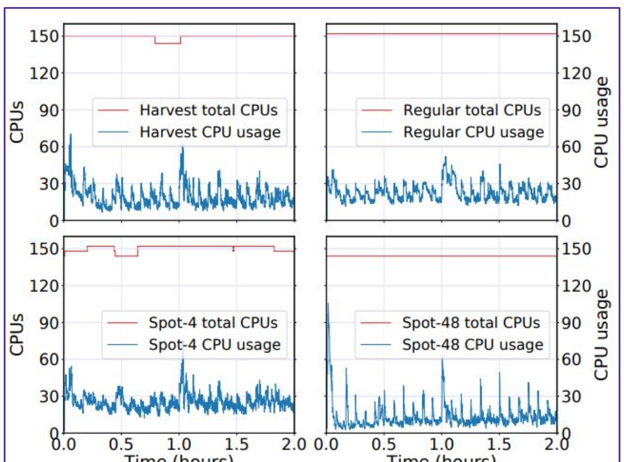
Figure 18: Harvest VMs vs Spot VMs. Hx refers to Harvest VMs with base size of x CPUs, and Sx refers to Spot VMs with x CPUs.

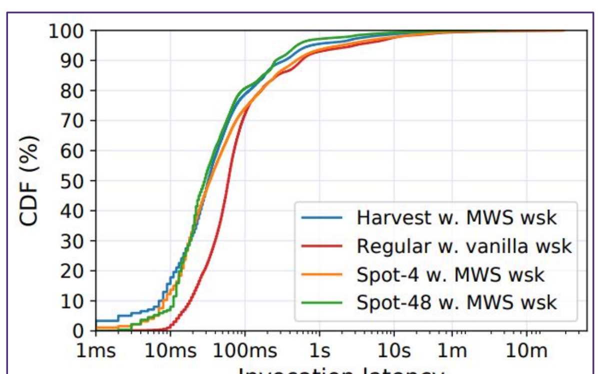
Evaluation
Experiments on Real Harvest VMs

- Implemented on Azure
- Number of CPUs cannot be controlled for these experiments
- Tested on four clusters shown in Table 4

VM type	Base CPUs	Max CPUs	Memory
Harvest	2	6	16GB
Regular	8	8	32GB
Spot-4	4	4	16GB
Spot-48	48	48	192GB

Table 4: Characteristics of the Harvest VMs, regular VMs, and Spot VMs used in the experiment in §7.6.





Author’s Conclusions

- Adopting harvested resources improves efficiency and reduces costs for FaaS applications
  - 48%-89% cost savings over dedicated resources
  - Only 4.1% of FaaS invocations are longer than 30 seconds and >90% of Harvest VMs live longer than 1 day
  - Resource variation is relatively stable with 70% of CPU change intervals longer than the longest invocation time
  - Eviction is rare and is a joint probability of long running invocations and an eviction occurring simultaneously
- MWS load balancing provides performance benefit of 22.6x higher throughput than vanilla OpenWhisk

Critique: Strengths

- A Harvest VM is more flexible and efficient than a spot instance
- Performance improvement serverless computing workloads on Harvest VMs significantly outperforms running them on regular VMs under the same cost budget
- Cold starts due to optimization of a load balancer are also minimal when the workload runs on Harvest VMs
- Cost savings when provisioned with the same amount of resources

Critique: Weaknesses

- Limited to small workloads (for example as FaaS), longer workloads could be evicted. Long applications (longer than 30s invocation period) should be run on a regular VM
- Resource variations in Harvest VMs – CPU changes. Despite offering a large amount of resources at low price, evictions and resource variation can impact the system reliability and performance
- Harvest VMs tend to be more heterogeneous than regular VMs

Critique: Evaluation

- Add a conclusion for every area of study (subtopic)
- Expand section 8 - Conclusion, briefly state conclusion for the study above

Gaps

- The latency of the three algorithms was depicted in a graph for regular VM (Faster and Cheaper Serverless Computing on Harvested Resources, p.733); however, it would be beneficial to build a graph for Harvest VM as well.
- For subtopic 7 add comparison for Harvest VMs vs Regular VMs
- Add strategy how to combine Regular and Harvest VMs

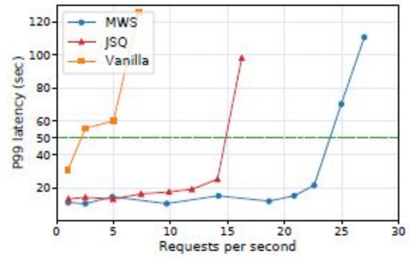


Figure 12: P99 latency across load balancing algorithms.

Questions

"If someone asks me what cloud computing is, I try not to get bogged down with definitions. I tell them that, simply put, cloud computing is a better way to run your business."

- Marc Benioff, Founder, CEO and Chairman, Salesforce