METRICS-DRIVEN SCALING PER AMAZON AURORA SERVERLESS: COME SI USA?

Amazon CloudWatch | AWS Lambda | Serverless

beSharp | 18 Ottobre 2019

L'avvento del paradigma **Serverless** ha dato una decisa svolta al processo di creazione di nuove infrastrutture Cloud e alla loro gestione.

Nel caso di architetture ospitate su Amazon Web Services, in particolare, è possibile ottenere in pochissimo tempo un'infrastruttura Serverless funzionante, **semplice da manutenere e con costi sensibilmente ridotti** rispetto ad un'architettura "classica" basata su server. Con un template di AWS CloudFormation anche basilare e dei semplici script di deploy, avremo ottenuto lo scheletro dell'infrastruttura. Sfruttando servizi come Amazon S3 e AWS Lambda, poi, avremo ottenuto storage e potenza di calcolo, senza necessità di manutenere alcun server.

Per rendere un'architettura completamente serverless, però, tutto ciò non è abbastanza: nel caso in cui occorra setuppare un sistema di Relational database management (RDBMS), la scelta del servizio migliore da utilizzare non è altrettanto scontata.

Il servizio AWS che meglio si presta a rendere serverless questo building block è senza dubbio **Aurora Serverless.**

Aurora serverless, rilasciato nell'agosto 2018, **un RDBMS relazionale completamente serverless** il cui vantaggio principale promesso è senza dubbio lo **scaling automatico.**

Ma è davvero una promessa mantenuta?

Andiamo nel dettaglio del suo funzionamento per rispondere finalmente a questa domanda.

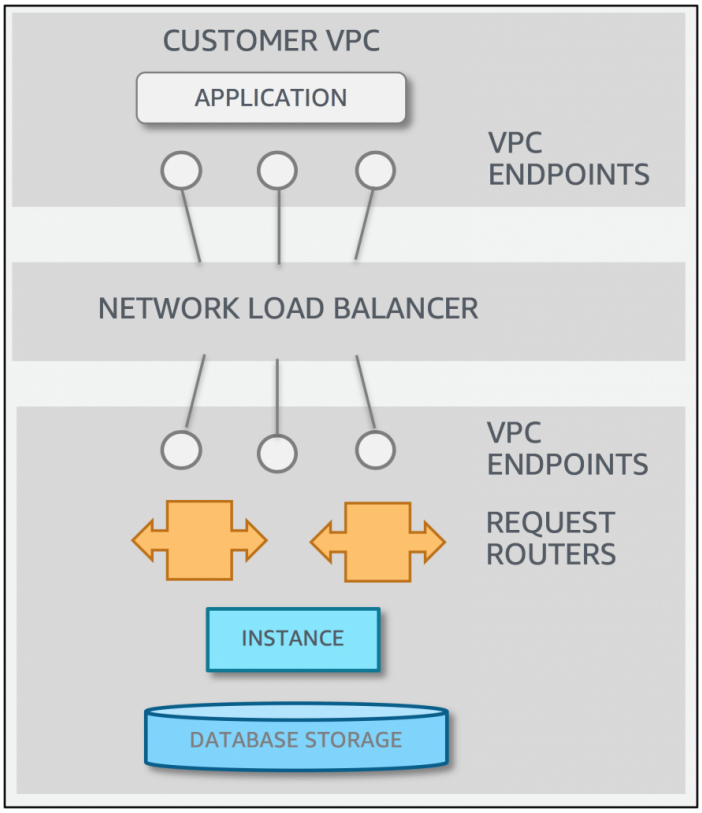
Cos'è Aurora serverless?

Amazon Aurora Serverless offre una configurazione on-demand di **auto-scaling per Amazon Aurora** (compatibile con MySQL e PostgreSQL) che prevede **start e spegnimento autonomi** dei database e **scale-up e down automatici** sulla base nelle richieste effettive delle applicazioni, il tutto senza

necessità di gestire alcuna istanza. Aurora Serverless è la soluzione adatta a fronteggiare anche **workloads intermittenti e non prevedibili.**

Vediamo come, entrando nel dettaglio della configurazione.

Come funziona l'auto-scaling di Aurora Serverless?



Lo scaling di Aurora Serverless interviene in modo automatico, triggerato ad esempio da metriche e allarmi. In questo modo **il develeloper è sollevato dalla necessità di progettare e gestire policy di scaling.**

Il servizio agisce creando **un volume di storage replicato su più Availabily Zones** (AZ). Verrà poi creato un **endpoint** all'interno della VPC a cui l'applicazione si connetterà. A questo punto, configurando dietro all'endpoint un **Network Load Balancer** (non visibile dall'utente) e un router per le richieste di tipo multi-tenant, Aurora Serverless comincerà ad indirizzare il traffico dei database alle istanze sottostanti.

Questa configurazione consentirà al cluster di scalare nel momento in cui ce ne sarà la necessità (o di riavviarsi dopo essere stato in pausa). Nel momento di picco, Aurora Serverless attingerà da **un pool di nodi già pronti e preconfigurati** aggiungendone un numero ottimale ai routers che stanno ricevendo il carico di richieste. Lo storage condiviso tra nodi permetterà ad Aurora Serverless di scalare istantaneamente sostenendo in pochissimi secondi qualsiasi workloads.

Lo scale-up (o lo scale-down) del cluster interverrà sulla base di limiti di capacità nella **CPU**, di numero di **connessioni attive** o di memoria occupata o nel caso in cui vengano rilevate **performance issues.**

Da tener presente quando ci si affida all'auto-scaling di Aurora Serverless sono i **tempi di cooldown:** 15 minuti per poter effettuare un primo scale-down e 310 secondi per scalare nuovamente verso il basso.

Il processo appena descritto è automatico e gestito nella sua totalità da AWS. È ovviamente possibile settare **valori custom di minimo e massimo** che andranno a triggerare le azioni di auto-scaling secondo le nostre necessità.

Il funzionamento di questo meccanismo di scaling autonomo è dipendente dall'esistenza di uno **scaling point,** ovvero di un preciso punto nel tempo in cui il database potrà cominciare in tutta sicurezza (senza cioè interrompere operazioni in corso) la procedura di scaling. Per default, se nessuno scaling point viene impostato, il servizio non è in grado di procedere con lo scaling. L'insuccesso nella ricerca di tale parametro da parte di Aurora Serverless, infatti, non causerà nessun errore, né lo stop del servizio: l'operazione di ricerca andrà avanti per tutto il tempo in cui il cluster del DB sarà sotto stress, ma nessun miglioramente di performance sarà applicato.

In assenza di scaling point impostato, è comunque possibile triggerare lo scale-up di Aurora Serverless **forzando l'aumento di capacità.** Si tenga presente che, in questo caso, nessuna connessione in corso sarà preservata. Le connessioni al DB potrebbero essere interrotte in un momento non sicuro con conseguente corruzione delle operazioni. Tale forzatura, se pur utile per ottenere lo scaling istantaneo, va usata con attenzione e **limitatamente ad applicazioni resilienti alle interruzioni di connessione.**

Lo scaling automatico di Aurora Serverless può aiutarci a coprire la maggioranza dei casi d'uso. Tuttavia, esistono specifici scenari in cui l'auto-scaling gestito da AWS non permetterebbe ai DB delle nostre applicazioni di scalare in modo appropriato.

Cosa fare?

In questi casi possiamo "aiutare" il processo di auto-scaling sfruttando gli **insights** messi a disposizione dall'applicazione.

Vediamo quindi nel dettaglio come agire quando l'auto-scaling gestito di AWS non performa bene.

Il parametro "desired ACUs"

Mentre nel caso di picchi di traffico prevedibili il cluster può facilmente essere "pre-allertato" in modo da essere in grado di soddisfare efficacemente e senza errori tutte le richieste, nel caso di picchi consistenti improvvisi di traffico, l'autoscaling potrebbe non riuscire a rispondere in tempi adeguati. Questo potrebbe dipendere da **un settaggio non ottimale del valore ACUs del cluster.** Un valore troppo basso, infatti, impedirebbe uno scaling tempestivo.

Ma modificare manualmente e puntualmente il parametro "desired ACUs" è la miglior soluzione?

CloudWatch custom Metrics

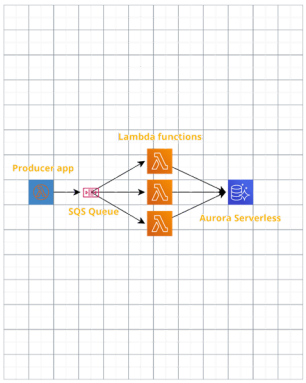
Quel che non tutti i developer sanno è che è possibile **rendere automatica la modifica del parametro "desired ACUs"** sfruttando semplicemente la raccolta di metriche di utilizzo dell'applicazione.

A venirci in aiuto in questo caso sono i servizi **Amazon CloudWatch** e **AWS Lambda.**

Raccogliendo le metriche dell'applicazione e mandandole a **CloudWatch custom metrics**, infatti, sarà possibile impostare allarmi col compito di triggerare **Lambda Function** all'interno del quale potremo implementare tutte le **logiche di scaling custom** che ci permetteranno di coprire potenzialmente qualunque necessità.

Use Case: un sistema in grado di elaborare e memorizzazione dati all'interno di un RDBMS a partire da un flusso di dati generato da un'applicazione.

Prima di concludere, ecco una soluzione realizzata con Amazon Aurora Serverless:



In questo progetto, ciò che il cliente voleva ottenere era un sistema efficace per elaborare e salvare dati all'interno di un RDBMS a partire da un flusso di dati generato da un'applicazione.

Il processo di analisi veniva azionato manualmente: il traffico verso il cluster, dunque, era pressoché nullo fino all'avvio di una nuova analisi. La soluzione iniziale faceva in modo che l'avvio dell'analisi triggerasse lo start di un numero consistente di funzioni Lambda col compito di leggere tutti i dati messi in coda dal servizio Amazon SQS, di elaborarli e di salvare tutti i risultati nel DB connettendosi al cluster. Queste connessioni al DB generavano un picco improvviso di traffico diretto al cluster che, a sua volta, provocava un'azione di auto-scaling. Tuttavia, questa situazione causava il fallimento di molte Lambda dovuto a numerosi errori di connessione dovuti ad all'azione di auto-scaling non sufficientemente rapida.

Per risolvere questo problema abbiamo fatto in modo che fosse la stessa applicazione a scatenare un aumento delle risorse del DB. Inoltre, agendo sul **parametro "delay"** abbiamo configurato i messaggi sulla coda SQS, in modo da rendersi disponibili solo dopo un certo periodo di tempo. Questo accorgimento lasciava il tempo al DB di scalare, prima di procedere ad attivare tutte le connessioni.

Alla fine dei conti la soluzione si è dimostrata altamente performante.

La funzionalità di auto-scaling fa fronte in modo efficace al workload consistente causato dallo start del task di analisi, per poi riportare il DB alla capacità di minimo iniziale. Performance e costi risultano quindi ottimizzati al 100%.

Con Aurora Serverless AWS è riuscita efficacemente a coprire anche l'ultimo gap rimasto nella creazione di soluzioni totalmente Serverless.

In questo articolo vi abbiamo dato qualche consiglio per sfruttarne al meglio tutte le potenzialità.

Ora è il momento di provarlo anche per voi!

Siamo curiosi di sapere cosa ne pensate

Arrivederci al prossimo articolo!

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it