OTTIMIZZAZIONE DEI COSTI DELLE APPLICAZIONI SERVERLESS.

Amazon API Gateway | Amazon CloudWatch | Amazon Kinesis Data Firehose | AWS Fargate | AWS Lambda | Kibana | Serverless

beSharp | 12 Giugno 2020

Siamo nel bel mezzo dell'**era serverless.** In un momento in cui i vantaggi di questo approccio diventano sempre più evidenti. Uno tra i più importanti per l'ascesa di questo paradigma è sicuramente la promessa di significativi **vantaggi economici.**

Ma è davvero così? Ancora più importante: stai sfruttando appieno il paradigma serverless?

Cos'è serverless?

Serverless è un paradigma nativo del Cloud che consente di spostare sempre più responsabilità operative verso il provider cloud, in questo caso AWS, aumentando **l'agilità** e **l'innovazione**. Serverless consente di creare ed eseguire applicazioni e servizi **senza pensare ai server fisici** e quindi eliminando le attività di gestione dell'infrastruttura come provisioning di server o cluster, patch, manutenzione del sistema operativo e provisioning della capacità. Serverless ti aiuta a eseguire e ridimensionare l'applicazione mentre **l'alta affidabilità** è fornita in modo trasparente dai servizi impiegati.

Durante la progettazione dell'architettura di un'applicazione serverless, è necessario pensare a diversi ambiti, tra cui **archiviazione**, **database**, **elaborazione**, **messaggistica** e molti altri.

AWS offre numerosi servizi che possono essere impiegati per creare un'applicazione serverless valida. Dal momento che è necessario scegliere lo strumento giusto per ogni ambito, la conoscenza, almeno ad alto livello, di ogni servizio è vitale per fare una scelta saggia.

Non solo Lambda

Quando si pensa a serverless, il primo servizio AWS che viene in mente è sicuramente Lambda, infatti la parte di **computing** di un'applicazione serverless è probabilmente la più iconica ed

importante. Tuttavia, un'architettura serverless non è composta solo da lambda, ma anche da diversi **servizi AWS gestiti** che a loro volta devono essere serverless.

Ecco un breve e non esaustivo elenco di servizi serverless, completamente gestiti, per ogni tier applicativo.

Computing

Naturalmente abbiamo AWS Lambda e Lambda@edge, che sono solitamente associate a serverless ed attraggono la maggior parte dei framework e delle soluzioni di terze parti, ma anche AWS Fargate, che è un container orchestrator serverless e fully managed costruito appositamente per docker container.

Archiviazione

Amazon Simple Storage Service (Amazon S3) fornisce storage a livello oggetti sicuro, durevole e altamente scalabile.

Database

Amazon DynamoDB è un servizio di database NoSQL veloce e flessibile per tutte le applicazioni che richiedono una latenza costante e bassa ad ogni dimensione.

Amazon Aurora Serverless è una configurazione su richiesta con scalabilità automatica per Amazon Aurora. Il database verrà avviato, arrestato e ridimensionato automaticamente la capacità in base alle esigenze dell'applicazione.

Proxy / API

Amazon API Gateway è un servizio completamente gestito e offre una piattaforma completa per la gestione delle API con supporto per l'autorizzazione, controllo accessi, monitoraggio e gestione delle versioni API.

Integrazione delle applicazioni

Amazon SNS è un servizio di pub / sub messaggistica completamente gestito che semplifica il disaccoppiamento e il ridimensionamento di microservizi, sistemi distribuiti e applicazioni serverless.

Amazon SQS è un servizio di accodamento messaggi completamente gestito che semplifica il disaccoppiamento e il ridimensionamento di microservizi, sistemi distribuiti e applicazioni serverless.

AWS AppSync semplifica lo sviluppo di applicazioni consentendo di creare un'API GraphQL flessibile per accedere, manipolare e combinare in modo sicuro i dati da una o più fonti.

Orchestration

AWS Step Functions semplifica il coordinamento dei componenti di applicazioni distribuite e microservizi mediante flussi di lavoro visivi. Step Functions è un modo affidabile per coordinare i componenti il flusso di funzioni lambda dell'applicazione.

Analytics

Amazon Kinesis è una piattaforma per lo streaming di dati su AWS, che offre potenti servizi per facilitare il caricamento e l'analisi dei dati di streaming e offre anche la possibilità di creare applicazioni di dati di streaming personalizzate per esigenze specializzate.

Amazon Athena è un servizio di query interattivo che semplifica l'analisi dei dati in Amazon S3 utilizzando SQL standard. Athena è serverless, quindi non c'è alcuna infrastruttura da gestire e paghi solo per le query che esegui.

AWS Lambda: pricing model e insidie

Quando si crea un'applicazione serverless, è importante tenere presente il suo **modello di pricing** al fine di strutturare i servizi e i trigger in un modo che consenta di ottenere **le migliori prestazioni al prezzo più basso**.

AWS Lambda ha un pricing model davvero interessante se usato nel modo giusto; inoltre, puoi sfruttare un generoso **free tier** di 1 milione di richieste gratuite al mese e 400.000 GB-seconds di tempo di elaborazione al mese. Questo free tier non scade dopo il primo anno.

Il modello dei prezzi Lambda è incentrato su **invocazioni e durata dell'esecuzione.**

Lambda conta una richiesta ogni volta che inizia l'esecuzione in risposta a una notifica di evento o invocazione di una funzione. La durata viene calcolata dal momento in cui l'esecuzione del codice inizia al momento in cui termina, arrotondata per eccesso ai 100 ms più vicini. Ciò significa che il tempo di esecuzione viene pagato "a scatti" di 100 ms. Avere una funzione il cui tempo di esecuzione medio è molto vicino alla soglia può quindi aumentare considerevolmente il costo complessivo di esecuzione.

Il prezzo di ogni "scatto" di 100 ms dipende dalla quantità di memoria allocata alla funzione. Un aumento delle dimensioni della memoria provoca un aumento equivalente della potenza di calcolo disponibile per la funzione.

Richieste: 0,20$ per 1 milione di richieste

Memoria (MB)	Prezzo per 100 ms
128	$0.0000002083
256	$0.0000008333
1024	$0.0000016667
1536	$0.0000025000
2048	$0.0000033333
3008	$0.0000048958

I seguenti fatti possono quindi essere dedotti dal modello di prezzo:

- L'ottimizzazione del **tempo di esecuzione** è cruciale, il codice deve usare le risorse disponibili al massimo e terminare nel minor tempo possibile.
- **La taglia della funzione Lambda** influisce sui tempi di risposta e sui costi complessivi. Occorre bilanciare il costo con i tempi di risposta, tenendo presente che il prezzo è per uno scatto di 100ms, per cui le funzioni con un tempo di esecuzione vicino alla soglia dello scatto sono quelle da ottimizzare maggiormente.
- Una funzione lambda non dovrebbe mai essere in attesa di **eventi esterni**: trigger, code, step functions dovrebbero essere utilizzati per disaccoppiare i servizi e invocare Lambda solo quando necessario.

Un altro inconveniente comune è quello di elaborare i messaggi da SQS in lotti 1 messaggio. Se la coda è piena, è bene utilizzare la stessa chiamata per elaborare il maggior numero possibile di messaggi poiché ogni chiamata ha un tempo di avvio e un tempo di caricamento che si sommano abbastanza rapidamente se si elabora solo 1 messaggio alla volta.

AWS API Gateway

API Gateway offre molte funzionalità, che possono essere preziose per **accelerare lo sviluppo** di applicazioni complesse ed è spesso considerata la soluzione naturale per i back-end serverless basati su Lambda. Tuttavia, il prezzo di ciascuna chiamata API è da 5 a 15 volte il prezzo di una chiamata lambda e mentre questo prezzo è adeguato per le funzionalità del servizio (ad es. gestione dell'autenticazione, memorizzazione nella cache, trasformazione richiesta / risposta, proxy), è facile capire che se queste funzionalità non vengono utilizzate, Api Gateway non è probabilmente la scelta migliore.

Esistono diverse situazioni in cui API Gateway viene utilizzato solo come proxy trasparente per passare l'intera richiesta HTTP a una funzione Lambda che quindi gestisce l'autenticazione, l'autorizzazione, l'analisi e la generazione della risposta. In questi casi, tuttavia ci sono spesso soluzioni più economiche: per esempio, un Elastic Load Balancer può anche essere utilizzato per inoltrare richieste HTTP a Lambda, e può essere più conveniente rispetto ad API Gateway per volumi elevati di invocazioni.

Quando il front-end è l'unico dei consumatori delle API di back-end, è possibile sfruttare Cognito per ottenere le credenziali IAM per gli utenti. Questi potranno poi invocare le funzioni lambda in modo sicuro utilizzando l'AWS Javascript SDK e le credenziali temporanee. Questa soluzione ti consente di ottenere sia l'autenticazione che l'autorizzazione tramite IAM, evitando la necessità di un gateway API al costo di funzionalità ridotte (devi preparare la tua richiesta sul frontend e gestire anche le risposte e gli errori nel tuo codice su lambda). Inoltre, non è possibile proteggere il back-end con un WAF, quindi questa strada potrebbe non essere sempre percorribile.

In sostanza, o si sfruttano le funzionalità di alto livello di Api Gateway oppure è meglio evitarlo per evitare di sprecare denaro.

Sistema di logging distribuito

Il logging è un'altra area in cui è possibile risparmiare denaro senza perdere funzionalità.

Per impostazione predefinita, viene utilizzato **CloudWatch Logs** per raccogliere i log di tutte le chiamate delle funzioni lambda.

La soluzione è perfetta per iniziare e il costo aumenta in proporzione al numero di GB di log generati, a partire da 0$.

Tuttavia, il costo per GB è piuttosto elevato e potresti essere sorpreso di vedere quanti GB di log sono generati da una tipica applicazione web serverless. Vi sono quindi casi in cui il costo di questo sistema di logging può diventare eccessivo e difficile da giustificare.

Quindi, come è possibile ottimizzare i costi di logging?

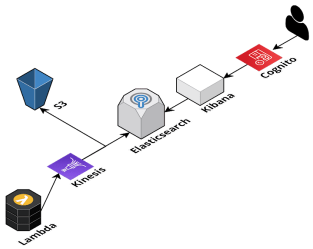
Innanzitutto, è necessario ridurre la quantità di log generati il più possibile, è quasi sempre conveniente riassumere le stesse informazioni in un formato più compatto.

È sempre meglio evitare messaggi lunghi e verbosi e registrare solo checkpoint o eventi importanti durante l'esecuzione, preferibilmente solo poche volte durante una singola chiamata.

Si consiglia inoltre di salvare i messaggi di log in un formato strutturato, come JSON, e di includere in un singolo messaggio tutte le informazioni necessarie per ricostruire l'evento che lo ha causato.

Ci sono anche casi in cui CloudWatch Logs è semplicemente non abbastanza avanzato da consentire un'adeguata analisi di problemi o situazioni eccezionali. In questi casi, è possibile beneficiare di un'infrastruttura di logging creata con Kinesis Firehose per lo lo streaming dei log, ElasticSearch e Kibana per la consultazione.

Questa soluzione ha un costo fisso piuttosto elevato, ma che tende a rimanere costante e in alcuni casi (quando la quantità di messaggi di log è stabile e alta) può essere molto inferiore al costo generato dai log di CloudWatch.



Questi sono solo alcuni dei trucchi che possono essere utilizzati per **ottimizzare i costi di un’applicazione serverless**, ovviamente ci sono molti più casi particolari e scenari specifici che possono beneficiare delle più disparate personalizzazioni infrastrutturali.

In generale su AWS è importante prendere in considerazione il modello di pricing di ciascun servizio e costruire sia l’infrastruttura che l’ applicazione al fine di trarne vantaggio anziché pagare di più per utilizzare i servizi in modo non ottimale.

Non perdere i prossimi articoli per ulteriori suggerimenti sull’argomento!

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

Copyright © 2011-2021 by beSharp srl - P.IVA IT02415160189