Lambda at scale: Tips & Tricks per far scalare la tua applicazione Serverless

9 Dicembre 2022 - 7 min. read

AWS Lambda | Serverless

NOTA: Tutte le quotas riportate in questo articolo fanno riferimento alla data di scrittura dell'articolo, potrebbero cambiare nel corso del tempo.

Introduzione

Serverless, serverless e serverless.. Se siete frequentatori del nostro blog, questa parola vi sarà entrata assolutamente in testa!

Questa tecnologia permette di focalizzarsi sul codice applicativo dimenticandosi della gestione dell'infrastruttura e pagando solo la potenza computazionale effettivamente utilizzata.

Quando si parla di serverless computing, però, sono molti gli aspetti da tenere in considerazione per evitare spiacevoli sorprese (magari in produzione). Le nostre applicazioni non dovranno essere soltanto Cloud-native, ma anche Serverless-native.

In questo articolo vi daremo molti spunti e consigli da tenere in considerazione quando si sviluppa un'applicazione su Lambda at scale, in particolare quando si lavora con Lambda su grandi numeri.

1. Attenzione alla concurrency

Quando si lavora su Lambda, uno dei primi limiti da tenere a mente è la concorrenza massima per region, di default 1000. Questo vuol dire che potranno esserci al massimo

1000 esecuzioni di Lambda contemporanee per ciascuna region AWS. Questo limite è condiviso tra tutte le istanze di Lambda, essendo un vincolo a livello di region.

Se si hanno quindi più workload nella stessa region, potrebbe essere utile configurare delle *reserved concurrency* per ogni workload, in modo da settare un cap massimo per ognuna di essi ed evitare comportamenti di difficile analisi e risoluzione.

In questo modo, i vari workload non si "ruberanno" la disponibilità di Lambda se mai uno di questi dovesse scalare fortemente.

In molti casi potrebbe essere che il limite di 1000 non sia sufficiente. Per fortuna, questo limite è di tipo *soft*, quindi può essere aumentato direttamente dalla console di AWS fino a qualche migliaio. Ricordatevi che la molte volte bisogna "giustificare" ad AWS molti aumenti di quotas.

E se ancora non bastasse?

Potrebbe essere che l'organizzazione dei tuoi account AWS abbia bisogno di una piccola revisione, magari è necessario configurare opportunamente un'Organization AWS e creare account AWS per ogni workload e ogni ambiente.

Se voleste rinfrescarvi la memoria su termini come provisioned concurrency, reserved concurrency e cold start time, fate un salto in questo articolo in cui analizziamo nel dettagli questi aspetti.

Tutti i suggerimenti che troverete in questo articolo derivano dall'aver aumentato il limite di Lambda da 1000 a 10.000 per una nostra applicazione di produzione.
L'increase di per sé è molto semplice, ma comporta una serie di possibili side-effects.

2. Chiamate a servizi esterni (Amazon S3 e AWS Secrets Manager)

Molto probabilmente, la nostra applicazione dovrà chiamare dei servizi AWS esterni.

Ad esempio, tra la varie best practice (non solo di AWS) vige quella di non hardcodare dati sensibili nel codice applicativo, ma di salvarli su servizi opportuni, magari cifrati *at rest* e *in transit* gestendo in maniera granulare le policy di accesso.

Parliamo ad esempio di AWS Secrets Manager e Parameter Store, ma avete mai dato un'occhiata alle loro quotas?

Con Secrets Manager siamo abbastanza tranquilli, siccome la quotas per recuperare un segreto è di 10.000 al secondo. Per Parameter Store, il limite è invece di 40 al secondo. Il limite è ovviamente aumentabile, ma bisogna tenerne conto prima del rilascio in produzione.

Sapevate che è anche possibile cachare i valori dei nostri segreti? Ebbene si, grazie alle **Lambda Extension** è possibile estendere le funzionalità delle nostre Lambda per scopi di monitoraggio, osservabilità e governance. È possibile utilizzare estensioni sviluppate da altri partner per integrarsi direttamente con loro, ad esempio per inoltrarvi log applicativi.

Sono anche presenti software per Parameter Store e Secret Manager che ci permettono di cachare i vari parametri e segreti, senza che dobbiamo farlo noi nel codice della Lambda.

Vi interessa sapere di più su come funzionano le Lambda Extension? Non site i soli: presto un articolo dedicato!

Magari la nostra Lambda dovrà anche scaricare un file da S3. Vediamo quindi le quotas di questo servizio: per la scrittura e lettura dei dati ha limiti molto particolari, ovvero 3500 PUT/COPY/POST/DELETE e 5000 GET/HEAD richieste al secondo, **per partitioned prefix.**

Questo limite non è condiviso a livello di account o di region, ma vale per ogni singolo bucket. Se dovete scaricare un singolo file molte volte da Lambda (migliaia di volte in qualche secondo), potreste incorrere in questo limite. Ecco il nostro suggerimento: provate a copiare il file in partizioni diverse.

In generale è bene ricordarsi di prestare attenzione alle quotas di tutti i servizi con cui la nostra applicazione dovrà interagire.

3. File system & /tmp storage

E se dobbiamo accedere ad un file system durante la fase di computing su Lambda? Ad esempio nel caso in cui occorra scaricare un dato da S3 per analizzarne il contenuto? Per fortuna, su Lambda puoi utilizzare il mounting point /tmp per questo tipo di operazioni.

Ovviamente non è persistente; quando la Lambda ritorna “fredda” AWS penserà a svuotare quella porzione di storage affinchè sia libera per l’esecuzione successiva.

Questo limite è da poco aumentabile: dai precedenti 512 MB, fino a 10GB. Per necessità di più spazio, era comunque possibile agganciare un’EFS mettendo la Lambda in VPC.

Cosa Abbiamo scoperto? In fase di scaling rapido e massivo, non è detto che la Lambda abbia questa porzione di storage vuoto, assicuratevi quindi di farne pulizia qualora doveste farci operazioni, potreste trovarvi dati vecchi!

4. Database connection

Molto probabilmente la nostra Lambda non dovrà solo richiamare servizi esterni o scrivere qualcosa sullo storage, ma dovrà anche collegarsi ad un database per effettuare delle operazioni di lettura e scrittura. Avete mai provato ad aprire 5000 connessioni temporaneamente ad un database di produzione? Lasciamo le conseguenze alla vostra immaginazione!

Per evitare di sovraccaricare il database con operazioni di questo tipo vi consigliamo di utilizzare un pool di connessione e interfacciarvi con esso. Su AWS, esiste RDS Proxy, un servizio gestito per questo tipo di operazioni. Potete quindi dimenticarvi di gestire il connection pool con il database.

5. Serverless application at scale

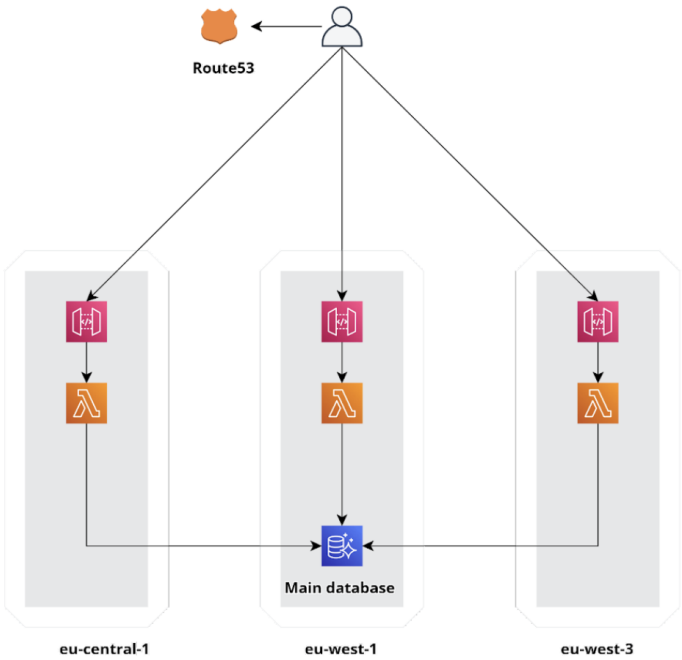
Fino ad ora abbiamo sottolineato l'importanza di valutate sempre in anticipo le metriche della vostra applicazione e controllate se vanno bene con quanto avete progettato.

E qualora non avete metriche pregresse da utilizzare, come nel nostro caso? Seguendo uno dei pillar del AWS Well Architect Framework, ovvero l'Operational Excellence, siamo riusciti a riadattare ed evolvere in tempi decisamente brevi la nuova infrastruttura.

Purtroppo i numeri avuti in produzione non potevano andare bene per un’infrastruttura rilasciata in una sola region. Abbiamo quindi sfruttato l’infrastruttura globale di AWS deployando l’infrastruttura in diverse Region e facendo load balancing tra di esse.

Aggiungiamo un po' di routing cross-region con Route 53, ed ecco che la nostra infrastruttura globale è pronta!

Sappiate quindi che se il vostro workload supera i limiti in una sola region, potrebbe essere giunta l'ora di diventare Global! Per fortuna su AWS è molto semplice. Se abbiamo poi generato l'infrastruttura seguendo il paradigma dell'Infastructure-as-Code (IaC) sarà questione di qualche click!



6. News da reinvent 2022: SnapStart

Siamo tornati da poco dal AWS re:Invent 2022 (tranquilli, uscirà un articolo a riguardo a breve!) dove hanno annunciato una funzionalità molto interessante per Lambda: SnapStart.

Questa feature è disponibile solo per Java, siccome mira a risolvere i problemi noti del cold start time elevato per questo engine.

Il funzionamento è abbastanza semplice: ad ogni versione della Lambda ne viene fatto uno snapshot, ovvero una sorta di AMI per Lambda, che verrà utilizzata nelle successive esecuzioni.

E' quindi possibile utilizzare degli hook appositi per istanziare il client dell'SDK di AWS, il cui tempo è noto per Java, e magari salvarsi alcuni segreti o informazioni che normalmente verrebbero raccolti in fase di startup della Lambda.

Conclusioni

AWS ci offre un'infrastruttura globale unica in termini di funzionalità e affidabilità, oltre ad una serie di "building blocks" per poter creare delle infrastrutture performanti e cost-effective, anche Serverless.

Ricordiamoci però di progettare il tutto affinchè l'applicazione sia scalabile anche su grandi numeri, qualora sia necessario.

Avete altri suggerimenti? Fatecelo sapere nei commenti!

Ci vediamo tra 14 giorni con un nuovo articolo su Proud2beCloud!

About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

Alessandro Bertini

DevOps Engineer @ beSharp, mi occupo di sviluppo software Cloud-native, fortemente orientato al paradigma Serverless!Appassionato di giochi da tavolo e videogame (come ogni buon smanettone!)
