Le migliori pratiche per Confluent Kafka

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Sommario

Le migliori pratiche per Confluent Kafka

TR-4912: Linee guida sulle best practice per l'archiviazione a livelli Confluent Kafka con NetApp

Karthikeyan Nagalingam, Joseph Kandatilparambil, NetApp Rankesh Kumar, Confluent

Apache Kafka è una piattaforma di streaming di eventi distribuita dalla comunità, in grado di gestire migliaia di miliardi di eventi al giorno. Inizialmente concepito come una coda di messaggistica, Kafka si basa su un'astrazione di un registro di commit distribuito. Da quando è stato creato e reso open source da LinkedIn nel 2011, Kafka si è evoluto da una coda di messaggi a una piattaforma di streaming di eventi a tutti gli effetti. Confluent fornisce la distribuzione di Apache Kafka tramite la piattaforma Confluent. La piattaforma Confluent integra Kafka con funzionalità aggiuntive per la comunità e commerciali, progettate per migliorare l'esperienza di streaming sia degli operatori che degli sviluppatori in produzione su larga scala.

Questo documento descrive le linee guida sulle best practice per l'utilizzo di Confluent Tiered Storage su un'offerta di storage a oggetti di NetApp, fornendo i seguenti contenuti:

- Verifica confluente con NetApp Object Storage – NetApp StorageGRID
- Test delle prestazioni di archiviazione a livelli
- Linee guida sulle best practice per Confluent sui sistemi di storage NetApp

Perché Confluent Tiered Storage?

Confluent è diventata la piattaforma di streaming in tempo reale predefinita per molte applicazioni, in particolare per i carichi di lavoro di big data, analisi e streaming. Tiered Storage consente agli utenti di separare l'elaborazione dall'archiviazione nella piattaforma Confluent. Rende l'archiviazione dei dati più conveniente, consente di archiviare quantità di dati praticamente infinite e di aumentare (o diminuire) i carichi di lavoro su richiesta, semplificando inoltre le attività amministrative come il ribilanciamento dei dati e dei tenant. I sistemi di archiviazione compatibili con S3 possono sfruttare tutte queste funzionalità per democratizzare i dati con tutti gli eventi in un unico posto, eliminando la necessità di una complessa ingegneria dei dati. Per maggiori informazioni sul motivo per cui dovresti utilizzare l'archiviazione a livelli per Kafka, controlla"questo articolo di Confluent" .

NetApp instaclustr supporta anche Kafka con storage a livelli dalla versione 3.8.1. Per maggiori dettagli, consultare qui "instaclust utilizzando l'archiviazione a livelli Kafka"

Perché NetApp StorageGRID per l'archiviazione a livelli?

StorageGRID è una piattaforma di archiviazione di oggetti leader del settore di NetApp. StorageGRID è una soluzione di archiviazione basata su oggetti e definita dal software che supporta le API di oggetti standard del settore, tra cui l'API Amazon Simple Storage Service (S3). StorageGRID archivia e gestisce dati non strutturati su larga scala per fornire un archivio di oggetti sicuro e durevole. I contenuti vengono posizionati nel posto giusto, al momento giusto e sul livello di archiviazione corretto, ottimizzando i flussi di lavoro e riducendo i costi per i rich media distribuiti a livello globale.

Il principale elemento di differenziazione di StorageGRID è il suo motore di policy Information Lifecycle

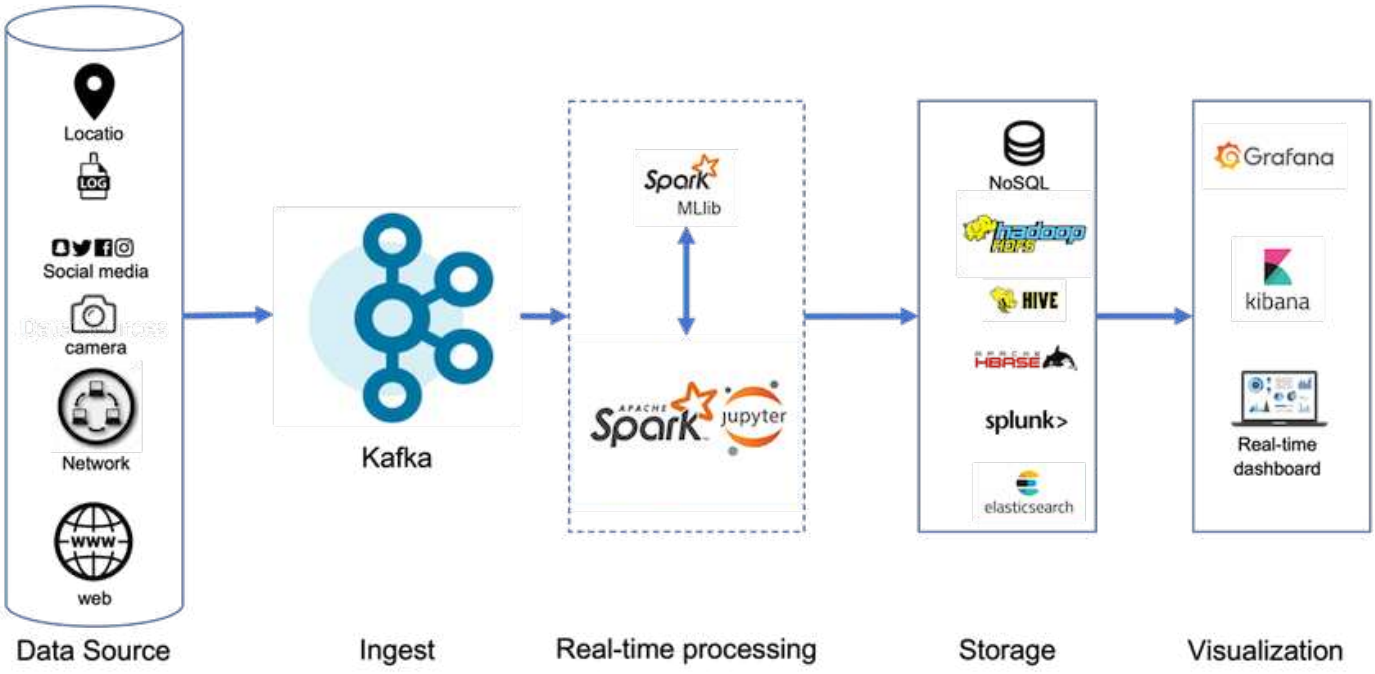
Management (ILM), che consente la gestione del ciclo di vita dei dati basata su policy. Il motore delle policy può utilizzare i metadati per gestire il modo in cui i dati vengono archiviati durante il loro ciclo di vita, ottimizzando inizialmente le prestazioni e ottimizzando automaticamente i costi e la durabilità man mano che i dati invecchiano.

Abilitazione dell'archiviazione a livelli confluenti

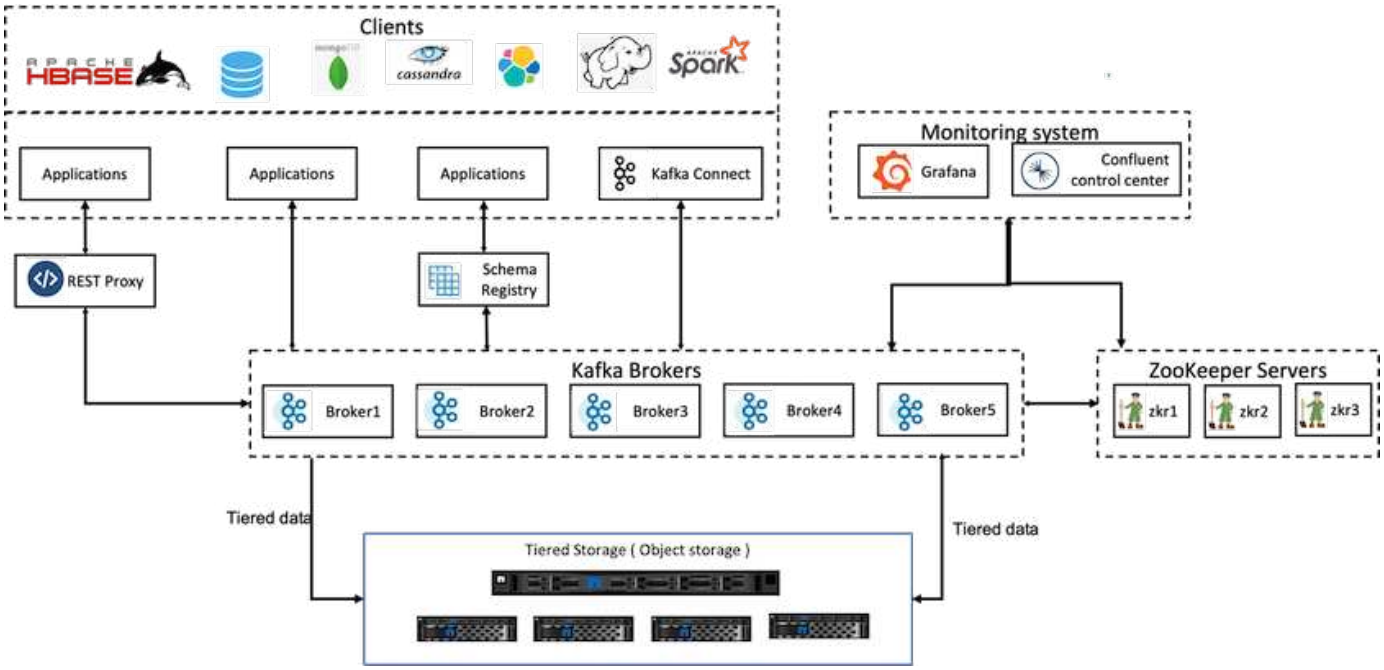
L'idea di base dell'archiviazione a livelli è quella di separare le attività di archiviazione dei dati da quelle di elaborazione degli stessi. Grazie a questa separazione, diventa molto più semplice per il livello di archiviazione dei dati e per il livello di elaborazione dei dati scalare in modo indipendente.

Una soluzione di archiviazione a livelli per Confluent deve tenere conto di due fattori. Innanzitutto, deve aggirare o evitare le proprietà comuni di coerenza e disponibilità dell'archivio oggetti, come incongruenze nelle operazioni LIST e occasionali indisponibilità degli oggetti. In secondo luogo, deve gestire correttamente l'interazione tra l'archiviazione a livelli e il modello di replicazione e tolleranza agli errori di Kafka, inclusa la possibilità che i leader zombie continuino a suddividere gli intervalli di offset. L'archiviazione di oggetti NetApp garantisce sia la disponibilità costante degli oggetti sia il modello HA che rende l'archiviazione obsoleta disponibile per intervalli di offset di livello. Lo storage di oggetti NetApp garantisce una disponibilità costante degli oggetti e un modello HA per rendere lo storage obsoleto disponibile per intervalli di offset di livello.

Con l'archiviazione a livelli, puoi utilizzare piattaforme ad alte prestazioni per letture e scritture a bassa latenza in prossimità della coda dei dati in streaming, e puoi anche utilizzare archivi di oggetti più economici e scalabili come NetApp StorageGRID per letture storiche ad alta velocità. Disponiamo anche di una soluzione tecnica per Spark con il controller di archiviazione NetApp; i dettagli sono disponibili qui. La figura seguente mostra come Kafka si inserisce in una pipeline di analisi in tempo reale.



La figura seguente illustra come NetApp StorageGRID si inserisce come livello di archiviazione degli oggetti di Confluent Kafka.



Dettagli dell'architettura della soluzione

Questa sezione riguarda l'hardware e il software utilizzati per la verifica Confluent. Queste informazioni sono applicabili alla distribuzione della piattaforma Confluent con storage NetApp . La tabella seguente illustra l'architettura della soluzione testata e i componenti di base.

Componenti della soluzione	Dettagli
Confluent Kafka versione 6.2	• Tre guardiani dello zoo • Cinque server broker • Cinque server di strumenti • Una Grafana • Un centro di controllo
Linux (Ubuntu 18.04)	Tutti i server
NetApp StorageGRID per l'archiviazione a livelli	• Software StorageGRID • 1 x SG1000 (bilanciatore di carico) • 4 x SGF6024 • 4 x 24 x 800 SSD • Protocollo S3 • 4 x 100 GbE (connettività di rete tra broker e istanze StorageGRID)
15 server Fujitsu PRIMERGY RX2540	Ciascuno dotato di: * 2 CPU, 16 core fisici totali * Intel Xeon * 256 GB di memoria fisica * Doppia porta 100 GbE

Panoramica della tecnologia

Questa sezione descrive la tecnologia utilizzata in questa soluzione.

NetApp StorageGRID

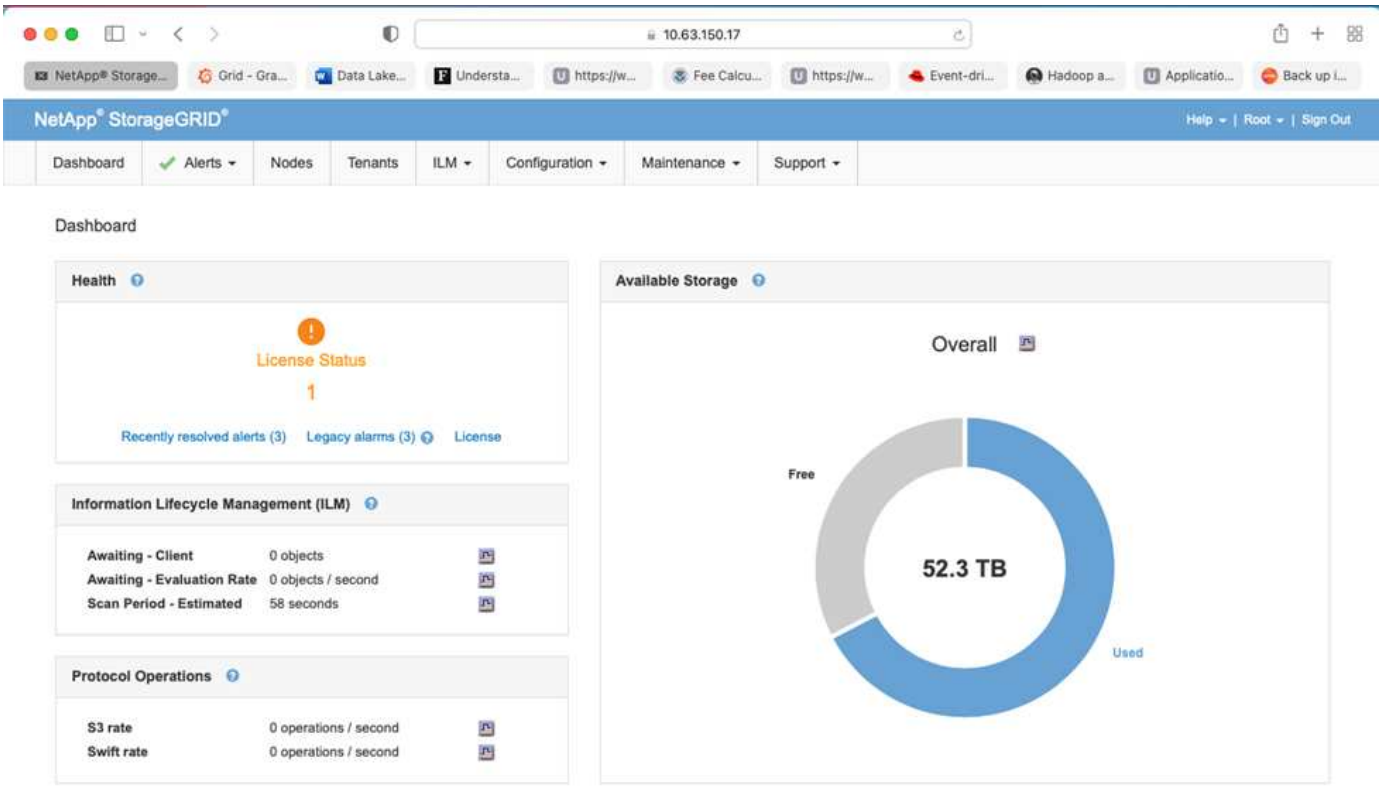
NetApp StorageGRID è una piattaforma di storage di oggetti conveniente e ad alte prestazioni. Utilizzando l'archiviazione a livelli, la maggior parte dei dati su Confluent Kafka, archiviati nell'archiviazione locale o nell'archiviazione SAN del broker, vengono scaricati nell'archivio oggetti remoto. Questa configurazione comporta notevoli miglioramenti operativi riducendo i tempi e i costi per ribilanciare, espandere o ridurre i cluster o sostituire un broker guasto. L'archiviazione di oggetti svolge un ruolo importante nella gestione dei dati che risiedono nel livello di archiviazione di oggetti, motivo per cui è importante scegliere l'archiviazione di oggetti giusta.

StorageGRID offre una gestione intelligente dei dati globali basata su policy, utilizzando un'architettura grid distribuita basata su nodi. Semplifica la gestione di petabyte di dati non strutturati e miliardi di oggetti attraverso il suo onnipresente spazio dei nomi degli oggetti globali combinato con sofisticate funzionalità di gestione dei dati. L'accesso agli oggetti con una sola chiamata si estende su più siti e semplifica le architetture ad alta disponibilità, garantendo al contempo un accesso continuo agli oggetti, indipendentemente dalle interruzioni del sito o dell'infrastruttura.

La multitenancy consente di gestire in modo sicuro più applicazioni cloud e dati aziendali non strutturati all'interno della stessa griglia, aumentando il ROI e i casi d'uso di NetApp StorageGRID. È possibile creare più livelli di servizio con policy del ciclo di vita degli oggetti basate sui metadati, ottimizzando la durabilità, la protezione, le prestazioni e la località in più aree geografiche. Gli utenti possono adattare le policy di gestione dei dati e monitorare e applicare limiti di traffico per riallinearsi al panorama dei dati senza interruzioni, man mano che le loro esigenze cambiano in ambienti IT in continua evoluzione.

Gestione semplice con Grid Manager

StorageGRID Grid Manager è un'interfaccia grafica basata su browser che consente di configurare, gestire e monitorare il sistema StorageGRID in sedi distribuite a livello globale da un unico pannello di controllo.



Con l'interfaccia di StorageGRID Grid Manager è possibile eseguire le seguenti attività:

- Gestisci repository di oggetti quali immagini, video e record distribuiti a livello globale, su scala petabyte.
- Monitorare i nodi e i servizi della griglia per garantire la disponibilità degli oggetti.
- Gestire il posizionamento dei dati degli oggetti nel tempo utilizzando le regole di gestione del ciclo di vita delle informazioni (ILM). Queste regole stabiliscono cosa accade ai dati di un oggetto dopo che sono stati acquisiti, come vengono protetti dalla perdita, dove vengono archiviati i dati dell'oggetto e per quanto tempo.
- Monitorare le transazioni, le prestazioni e le operazioni all'interno del sistema.

Politiche di gestione del ciclo di vita delle informazioni

StorageGRID dispone di policy di gestione dei dati flessibili che includono la conservazione di copie replicate degli oggetti e l'utilizzo di schemi EC (erasure coding) come 2+1 e 4+2 (tra gli altri) per archiviare gli oggetti, a seconda dei requisiti specifici di prestazioni e protezione dei dati. Poiché i carichi di lavoro e i requisiti cambiano nel tempo, è normale che anche le policy ILM debbano cambiare nel tempo. La modifica delle policy ILM è una funzionalità fondamentale che consente ai clienti StorageGRID di adattarsi in modo rapido e semplice al loro ambiente in continua evoluzione.

Prestazione

StorageGRID aumenta le prestazioni aggiungendo più nodi di storage, che possono essere VM, bare metal o appliance appositamente realizzate come"SG5712, SG5760, SG6060 o SGF6024" . Nei nostri test abbiamo superato i requisiti di prestazioni chiave di Apache Kafka con una griglia di tre nodi di dimensioni minime utilizzando l'appliance SGF6024. Man mano che i clienti ampliano il loro cluster Kafka con broker aggiuntivi, possono aggiungere più nodi di archiviazione per aumentare prestazioni e capacità.

Configurazione del bilanciatore del carico e degli endpoint

I nodi amministrativi in StorageGRID forniscono l'interfaccia utente (UI) di Grid Manager e l'endpoint API REST per visualizzare, configurare e gestire il sistema StorageGRID , nonché registri di controllo per monitorare l'attività del sistema. Per fornire un endpoint S3 ad alta disponibilità per l'archiviazione a livelli Confluent Kafka, abbiamo implementato il bilanciatore del carico StorageGRID , che viene eseguito come servizio sui nodi di amministrazione e sui nodi gateway. Inoltre, il bilanciatore del carico gestisce anche il traffico locale e comunica con il GSLB (Global Server Load Balancing) per facilitare il ripristino in caso di emergenza.

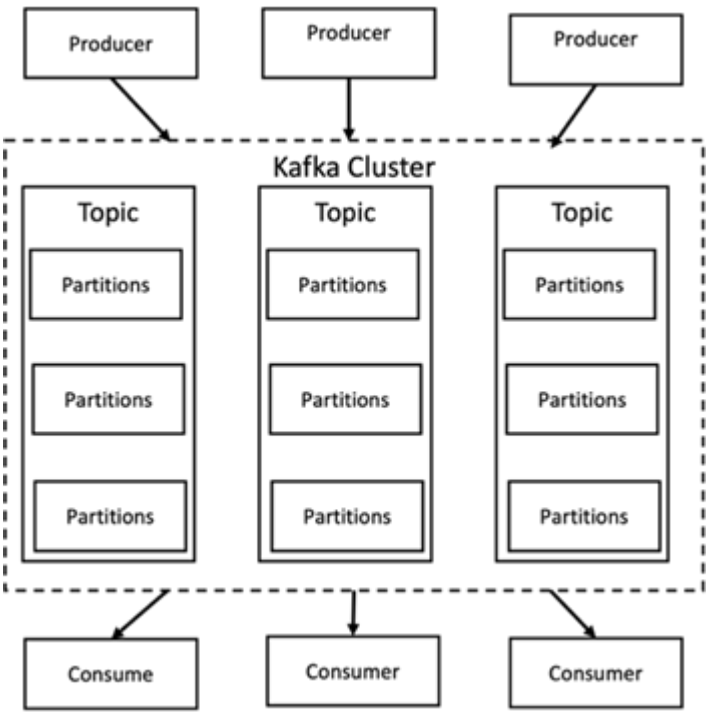
Per migliorare ulteriormente la configurazione degli endpoint, StorageGRID fornisce criteri di classificazione del traffico integrati nel nodo di amministrazione, consente di monitorare il traffico del carico di lavoro e applica vari limiti di qualità del servizio (QoS) ai carichi di lavoro. I criteri di classificazione del traffico vengono applicati agli endpoint del servizio StorageGRID Load Balancer per i nodi gateway e i nodi amministrativi. Queste politiche possono aiutare a modellare e monitorare il traffico.

Classificazione del traffico in StorageGRID

StorageGRID ha funzionalità QoS integrate. Le policy di classificazione del traffico possono aiutare a monitorare diversi tipi di traffico S3 provenienti da un'applicazione client. È quindi possibile creare e applicare policy per limitare questo traffico in base alla larghezza di banda in entrata/uscita, al numero di richieste di lettura/scrittura simultanee o alla velocità delle richieste di lettura/scrittura.

Apache Kafka

Apache Kafka è un framework di implementazione di un bus software che utilizza l'elaborazione di flussi, scritto in Java e Scala. Il suo scopo è fornire una piattaforma unificata, ad alta produttività e bassa latenza per la gestione di feed di dati in tempo reale. Kafka può connettersi a un sistema esterno per l'esportazione e l'importazione di dati tramite Kafka Connect e fornisce Kafka streams, una libreria di elaborazione di flussi Java. Kafka utilizza un protocollo binario basato su TCP, ottimizzato per l'efficienza e basato su un'astrazione di "set di messaggi" che raggruppa naturalmente i messaggi per ridurre il sovraccarico del roundtrip di rete. Ciò consente operazioni sequenziali su disco più grandi, pacchetti di rete più grandi e blocchi di memoria contigui, consentendo così a Kafka di trasformare un flusso continuo di scritture di messaggi casuali in scritture lineari. La figura seguente illustra il flusso di dati di base di Apache Kafka.



Kafka memorizza messaggi chiave-valore provenienti da un numero arbitrario di processi chiamati produttori. I dati possono essere suddivisi in diverse partizioni all'interno di argomenti diversi. All'interno di una partizione, i messaggi vengono ordinati rigorosamente in base ai loro offset (la posizione di un messaggio all'interno di una partizione) e indicizzati e archiviati insieme a un timestamp. Altri processi chiamati consumatori possono leggere i messaggi dalle partizioni. Per l'elaborazione dei flussi, Kafka offre l'API Streams che consente di scrivere applicazioni Java che utilizzano dati da Kafka e scrivono i risultati in Kafka. Apache Kafka funziona anche con sistemi di elaborazione di flussi esterni come Apache Apex, Apache Flink, Apache Spark, Apache Storm e Apache NiFi.

Kafka viene eseguito su un cluster di uno o più server (chiamati broker) e le partizioni di tutti gli argomenti sono distribuite tra i nodi del cluster. Inoltre, le partizioni vengono replicate su più broker. Questa architettura consente a Kafka di distribuire flussi massivi di messaggi in modalità fault-tolerant e gli ha permesso di sostituire alcuni dei sistemi di messaggistica convenzionali come Java Message Service (JMS), Advanced Message Queuing Protocol (AMQP) e così via. A partire dalla versione 0.11.0.0, Kafka offre scritture transazionali, che forniscono l'elaborazione di flussi esattamente una volta utilizzando l'API Streams.

Kafka supporta due tipi di argomenti: regolari e compattati. Gli argomenti regolari possono essere configurati con un limite di tempo di conservazione o di spazio. Se sono presenti record più vecchi del tempo di conservazione specificato o se viene superato il limite di spazio per una partizione, Kafka può eliminare i vecchi dati per liberare spazio di archiviazione. Per impostazione predefinita, gli argomenti sono configurati con un tempo di conservazione di 7 giorni, ma è anche possibile archiviare i dati a tempo indeterminato. Per gli argomenti compattati, i record non scadono in base a limiti di tempo o di spazio. Kafka, invece, tratta i messaggi successivi come aggiornamenti di messaggi più vecchi con la stessa chiave e garantisce di non eliminare mai l'ultimo messaggio per chiave. Gli utenti possono eliminare completamente i messaggi scrivendo un cosiddetto messaggio tombstone con il valore null per una chiave specifica.

Ci sono cinque API principali in Kafka:

- **API del produttore.** Consente a un'applicazione di pubblicare flussi di record.
- **API per i consumatori.** Consente a un'applicazione di iscriversi ad argomenti ed elaborare flussi di record.
- **API del connettore.** Esegue le API riutilizzabili del produttore e del consumatore che possono collegare gli argomenti alle applicazioni esistenti.
- **API Stream.** Questa API converte i flussi di input in output e produce il risultato.
- **API di amministrazione.** Utilizzato per gestire argomenti Kafka, broker e altri oggetti Kafka.

Le API consumer e producer si basano sul protocollo di messaggistica Kafka e offrono un'implementazione di riferimento per i client consumer e producer Kafka in Java. Il protocollo di messaggistica sottostante è un protocollo binario che gli sviluppatori possono utilizzare per scrivere i propri client consumer o producer in qualsiasi linguaggio di programmazione. Ciò sblocca Kafka dall'ecosistema Java Virtual Machine (JVM). Un elenco dei client non Java disponibili è disponibile nel wiki di Apache Kafka.

Casi d'uso di Apache Kafka

Apache Kafka è particolarmente diffuso per la messaggistica, il monitoraggio delle attività sui siti web, le metriche, l'aggregazione dei log, l'elaborazione dei flussi, l'event sourcing e la registrazione degli commit.

- Kafka ha migliorato la produttività, ha integrato il partizionamento, la replica e la tolleranza agli errori, il che lo rende una buona soluzione per applicazioni di elaborazione dei messaggi su larga scala.
- Kafka può ricostruire le attività di un utente (visualizzazioni di pagina, ricerche) in una pipeline di monitoraggio come un insieme di feed di pubblicazione-sottoscrizione in tempo reale.
- Kafka viene spesso utilizzato per i dati di monitoraggio operativo. Ciò comporta l'aggregazione di statistiche provenienti da applicazioni distribuite per produrre feed centralizzati di dati operativi.

- Molte persone utilizzano Kafka come sostituto di una soluzione di aggregazione dei log. L'aggregazione dei log in genere raccoglie i file di log fisici dai server e li colloca in un luogo centrale (ad esempio, un file server o HDFS) per l'elaborazione. Kafka astrae i dettagli dei file e fornisce un'astrazione più pulita dei dati di log o di eventi come flusso di messaggi. Ciò consente un'elaborazione a bassa latenza e un supporto più semplice per più fonti di dati e un consumo di dati distribuito.
- Molti utenti di Kafka elaborano i dati in pipeline di elaborazione costituite da più fasi, in cui i dati di input grezzi vengono utilizzati dagli argomenti di Kafka e quindi aggregati, arricchiti o altrimenti trasformati in nuovi argomenti per un ulteriore utilizzo o un'elaborazione successiva. Ad esempio, una pipeline di elaborazione per consigliare articoli di notizie potrebbe analizzare il contenuto degli articoli dai feed RSS e pubblicarlo in un argomento "articoli". Un'ulteriore elaborazione potrebbe normalizzare o deduplicare questo contenuto e pubblicare il contenuto dell'articolo ripulito in un nuovo argomento; una fase di elaborazione finale potrebbe tentare di consigliare questo contenuto agli utenti. Tali pipeline di elaborazione creano grafici di flussi di dati in tempo reale basati sui singoli argomenti.
- L'event souring è uno stile di progettazione delle applicazioni in cui le modifiche di stato vengono registrate come una sequenza di record ordinata nel tempo. Il supporto di Kafka per dati di log memorizzati di grandi dimensioni lo rende un backend eccellente per un'applicazione creata in questo stile.
- Kafka può fungere da una sorta di registro di commit esterno per un sistema distribuito. Il registro aiuta a replicare i dati tra i nodi e funge da meccanismo di risincronizzazione per i nodi non riusciti per ripristinare i propri dati. La funzionalità di compattazione dei log in Kafka aiuta a supportare questo caso d'uso.

Confluente

Confluent Platform è una piattaforma pronta per le aziende che completa Kafka con funzionalità avanzate progettate per accelerare lo sviluppo e la connettività delle applicazioni, abilitare le trasformazioni tramite l'elaborazione in streaming, semplificare le operazioni aziendali su larga scala e soddisfare rigorosi requisiti architettonici. Sviluppato dai creatori originali di Apache Kafka, Confluent amplia i vantaggi di Kafka con funzionalità di livello aziendale, eliminando al contempo l'onere della gestione o del monitoraggio di Kafka. Oggi, oltre l'80% delle aziende Fortune 100 si avvale della tecnologia di streaming dei dati e la maggior parte di queste utilizza Confluent.

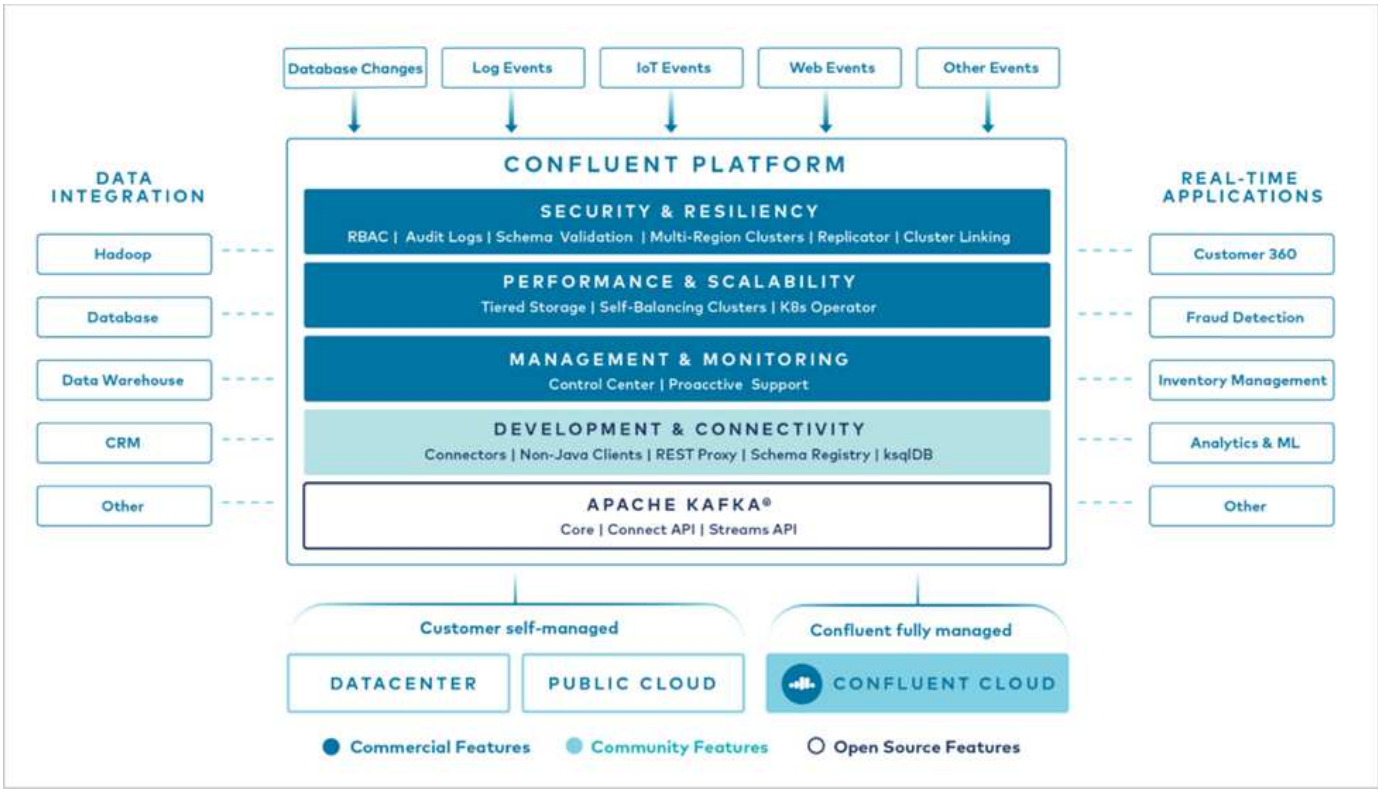
Perché Confluent?

Integrando dati storici e in tempo reale in un'unica fonte centrale di verità, Confluent semplifica la creazione di una categoria completamente nuova di applicazioni moderne basate sugli eventi, l'acquisizione di una pipeline di dati universale e lo sblocco di nuovi potenti casi d'uso con piena scalabilità, prestazioni e affidabilità.

A cosa serve Confluent?

Confluent Platform ti consente di concentrarti su come ricavare valore aziendale dai tuoi dati anziché preoccuparti dei meccanismi sottostanti, ad esempio come i dati vengono trasportati o integrati tra sistemi diversi. Nello specifico, Confluent Platform semplifica la connessione delle fonti di dati a Kafka, la creazione di applicazioni di streaming, nonché la protezione, il monitoraggio e la gestione dell'infrastruttura Kafka. Oggi, Confluent Platform viene utilizzata per un'ampia gamma di casi d'uso in numerosi settori, dai servizi finanziari, alla vendita al dettaglio omnicanale e alle auto autonome, fino al rilevamento delle frodi, ai microservizi e all'IoT.

La figura seguente mostra i componenti della piattaforma Confluent Kafka.



Panoramica della tecnologia di streaming degli eventi di Confluent

Il cuore della piattaforma Confluent è "Apache Kafka" , la piattaforma di streaming distribuita open source più popolare. Le principali capacità di Kafka sono le seguenti:

- Pubblica e abbonati a flussi di record.
- Memorizzare flussi di record in modo tollerante agli errori.
- Elaborare flussi di record.

Confluent Platform include anche Schema Registry, REST Proxy, un totale di oltre 100 connettori Kafka predefiniti e ksqlDB.

Panoramica delle funzionalità aziendali della piattaforma Confluent

- **Centro di controllo confluente.** Un sistema basato su GUI per la gestione e il monitoraggio di Kafka. Consente di gestire facilmente Kafka Connect e di creare, modificare e gestire connessioni ad altri sistemi.
- **Confluent per Kubernetes.** Confluent per Kubernetes è un operatore Kubernetes. Gli operatori Kubernetes estendono le capacità di orchestrazione di Kubernetes fornendo funzionalità e requisiti esclusivi per una specifica applicazione della piattaforma. Per Confluent Platform, ciò include una notevole semplificazione del processo di distribuzione di Kafka su Kubernetes e l'automazione delle tipiche attività del ciclo di vita dell'infrastruttura.
- **Connettori confluenti con Kafka.** I connettori utilizzano l'API Kafka Connect per connettere Kafka ad altri sistemi, quali database, archivi chiave-valore, indici di ricerca e file system. Confluent Hub dispone di connettori scaricabili per le fonti e i sink di dati più diffusi, comprese versioni completamente testate e supportate di questi connettori con Confluent Platform. Maggiori dettagli possono essere trovati "Qui" .
- **Cluster autobilancianti.** Fornisce bilanciamento automatico del carico, rilevamento degli errori e auto-riparazione. Fornisce supporto per l'aggiunta o la disattivazione di broker in base alle necessità, senza necessità di ottimizzazione manuale.

- **Collegamento di cluster confluenti.** Collega direttamente i cluster tra loro e rispecchia gli argomenti da un cluster all'altro tramite un ponte di collegamento. Il collegamento dei cluster semplifica la configurazione di distribuzioni multi-datacenter, multi-cluster e cloud ibride.
- **Bilanciatore automatico dei dati Confluent.** Monitora il cluster per quanto riguarda il numero di broker, la dimensione delle partizioni, il numero di partizioni e il numero di leader all'interno del cluster. Consente di spostare i dati per creare un carico di lavoro uniforme nel cluster, limitando al contempo il traffico di ribilanciamento per ridurre al minimo l'effetto sui carichi di lavoro di produzione durante il ribilanciamento.
- **Replicatore confluente.** Rende più semplice che mai la gestione di più cluster Kafka in più data center.
- **Archiviazione a livelli.** Offre opzioni per archiviare grandi volumi di dati Kafka utilizzando il tuo provider cloud preferito, riducendo così i costi e gli oneri operativi. Grazie all'archiviazione a livelli, puoi conservare i dati su un archivio oggetti conveniente e utilizzare broker di scalabilità solo quando hai bisogno di più risorse di elaborazione.
- **Client JMS confluente.** Confluent Platform include un client compatibile con JMS per Kafka. Questo client Kafka implementa l'API standard JMS 1.1, utilizzando i broker Kafka come backend. Questa funzionalità è utile se si dispone di applicazioni legacy che utilizzano JMS e si desidera sostituire il broker di messaggi JMS esistente con Kafka.
- **Proxy MQTT confluente.** Fornisce un modo per pubblicare dati direttamente su Kafka da dispositivi e gateway MQTT senza la necessità di un broker MQTT intermedio.
- **Plugin di sicurezza Confluent.** I plugin di sicurezza Confluent vengono utilizzati per aggiungere funzionalità di sicurezza a vari strumenti e prodotti della piattaforma Confluent. Attualmente è disponibile un plugin per il proxy REST Confluent che aiuta ad autenticare le richieste in arrivo e a propagare il principal autenticato alle richieste a Kafka. Ciò consente ai client proxy REST Confluent di utilizzare le funzionalità di sicurezza multitenant del broker Kafka.

Verifica confluente

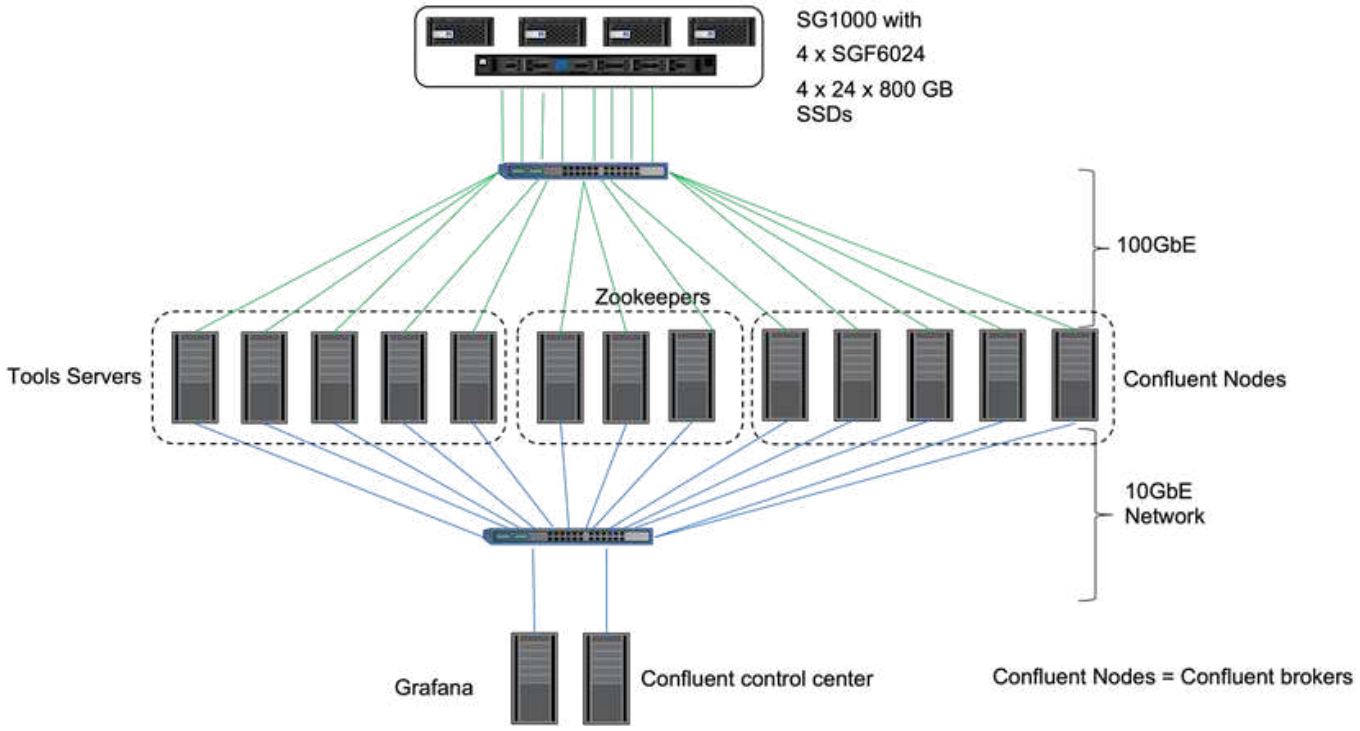
Abbiamo eseguito la verifica con Confluent Platform 6.2 Tiered Storage in NetApp StorageGRID. I team NetApp e Confluent hanno lavorato insieme a questa verifica ed eseguito i casi di test richiesti per la verifica.

Configurazione della piattaforma Confluent

Per la verifica abbiamo utilizzato la seguente configurazione.

Per la verifica, abbiamo utilizzato tre guardiani dello zoo, cinque broker, cinque server di esecuzione di script di test, server di strumenti denominati con 256 GB di RAM e 16 CPU. Per l'archiviazione NetApp , abbiamo utilizzato StorageGRID con un bilanciatore di carico SG1000 con quattro SGF6024. Lo storage e i broker erano collegati tramite connessioni 100GbE.

La figura seguente mostra la topologia di rete della configurazione utilizzata per la verifica Confluent.



I server degli strumenti agiscono come client applicativi che inviano richieste ai nodi Confluent.

Configurazione di archiviazione a livelli confluenti

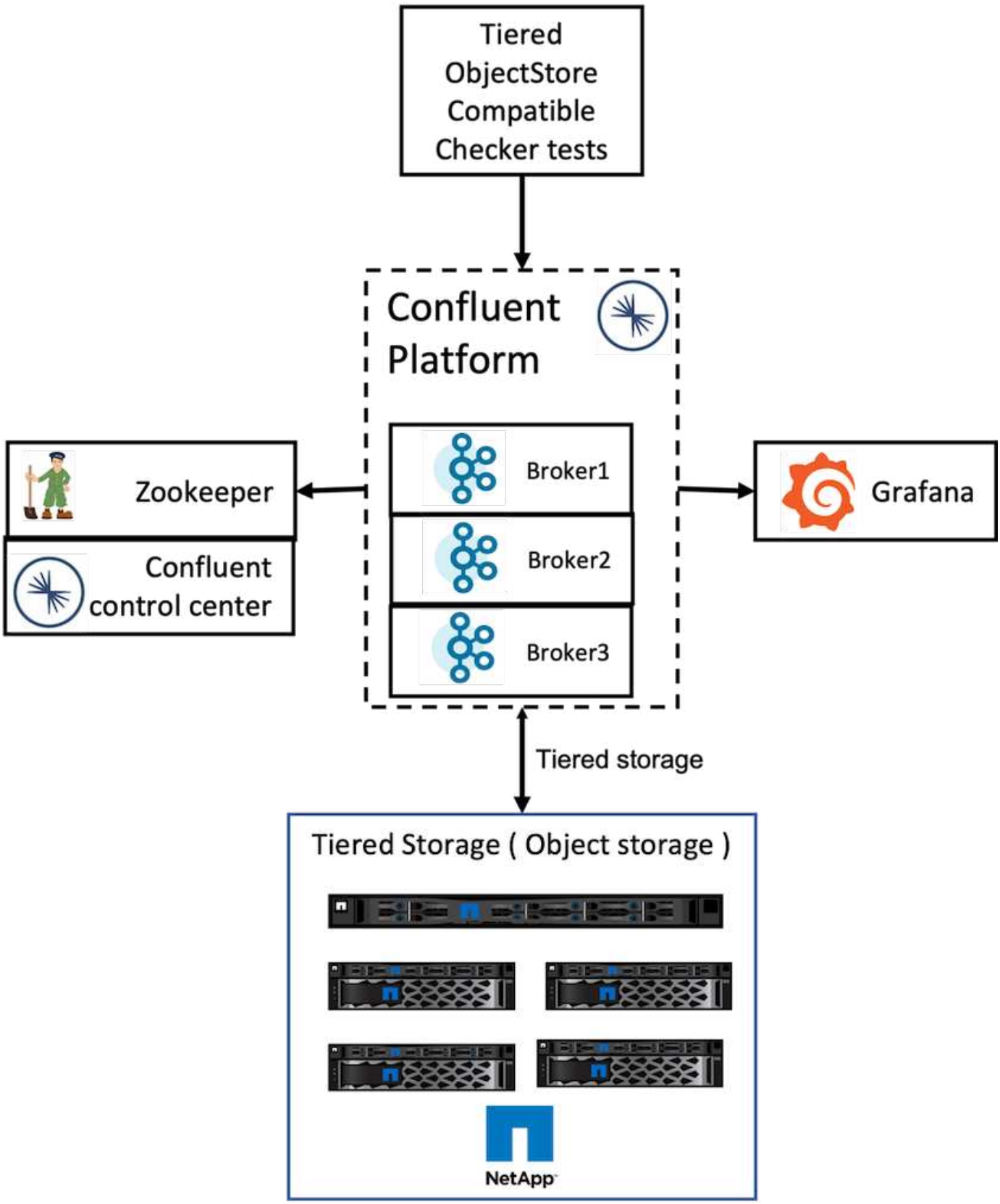
La configurazione dell'archiviazione a livelli richiede i seguenti parametri in Kafka:

```
Confluent.tier.archiver.num.threads=16
confluent.tier.fetcher.num.threads=32
confluent.tier.enable=true
confluent.tier.feature=true
confluent.tier.backend=S3
confluent.tier.s3.bucket=kafkasgdbucket1-2
confluent.tier.s3.region=us-west-2
confluent.tier.s3.cred.file.path=/data/kafka/.ssh/credentials
confluent.tier.s3.aws.endpoint.override=http://kafkasgd.rtpppe.netapp.com:
10444/
confluent.tier.s3.force.path.style.access=true
```

Per la verifica abbiamo utilizzato StorageGRID con il protocollo HTTP, ma funziona anche HTTPS. La chiave di accesso e la chiave segreta sono memorizzate nel nome del file fornito nel `confluent.tier.s3.cred.file.path` parametro.

Archiviazione di oggetti NetApp - StorageGRID

Abbiamo configurato la configurazione a sito singolo in StorageGRID per la verifica.



Test di verifica

Per la verifica abbiamo completato i seguenti cinque casi di test. Questi test vengono eseguiti sul framework Trogdor. I primi due erano test di funzionalità, mentre i restanti tre erano test di prestazioni.

Test di correttezza dell'archivio oggetti

Questo test determina se tutte le operazioni di base (ad esempio, get/put/delete) sull'API dell'archivio oggetti funzionano bene in base alle esigenze dell'archiviazione a livelli. Si tratta di un test di base che ogni servizio di archiviazione di oggetti dovrebbe aspettarsi di superare prima dei test successivi. È un test assertivo che può essere superato o fallito.

Test di correttezza della funzionalità di tiering

Questo test determina se la funzionalità di archiviazione a livelli end-to-end funziona bene con un test assertivo che può avere esito positivo o negativo. Il test crea un argomento di prova che per impostazione predefinita è configurato con il tiering abilitato e una dimensione dell'hotset molto ridotta. Produce un flusso di eventi per l'argomento di test appena creato, attende che i broker archivino i segmenti nell'archivio oggetti, quindi consuma il flusso di eventi e convalida che il flusso consumato corrisponda al flusso prodotto. Il numero di messaggi prodotti nel flusso di eventi è configurabile, il che consente all'utente di generare un carico di lavoro sufficientemente ampio in base alle esigenze di test. La dimensione ridotta dell'hotset garantisce che i recuperi dei consumatori al di fuori del segmento attivo vengano serviti solo dall'archivio oggetti; ciò aiuta a verificare la correttezza dell'archivio oggetti per le letture. Abbiamo eseguito questo test con e senza un'iniezione di errore nell'archivio oggetti. Abbiamo simulato un guasto del nodo arrestando il servizio di gestione dei servizi in uno dei nodi in StorageGRID e verificando che la funzionalità end-to-end funzioni con l'archiviazione di oggetti.

Benchmark di recupero dei livelli

Questo test ha convalidato le prestazioni di lettura dell'archiviazione di oggetti a livelli e ha controllato le richieste di lettura di recupero dell'intervallo sotto carico pesante dai segmenti generati dal benchmark. In questo benchmark, Confluent ha sviluppato client personalizzati per soddisfare le richieste di recupero dei livelli.

Benchmark del carico di lavoro produzione-consumo

Questo test ha generato indirettamente un carico di lavoro di scrittura sull'archivio oggetti tramite l'archiviazione dei segmenti. Il carico di lavoro di lettura (segmenti letti) è stato generato dall'archiviazione degli oggetti quando i gruppi di consumatori hanno recuperato i segmenti. Questo carico di lavoro è stato generato dallo script di test. Questo test ha verificato le prestazioni di lettura e scrittura sull'archiviazione di oggetti in thread paralleli. Abbiamo eseguito i test con e senza l'inserimento di errori nell'archivio oggetti, come abbiamo fatto per il test di correttezza della funzionalità di tiering.

Benchmark del carico di lavoro di conservazione

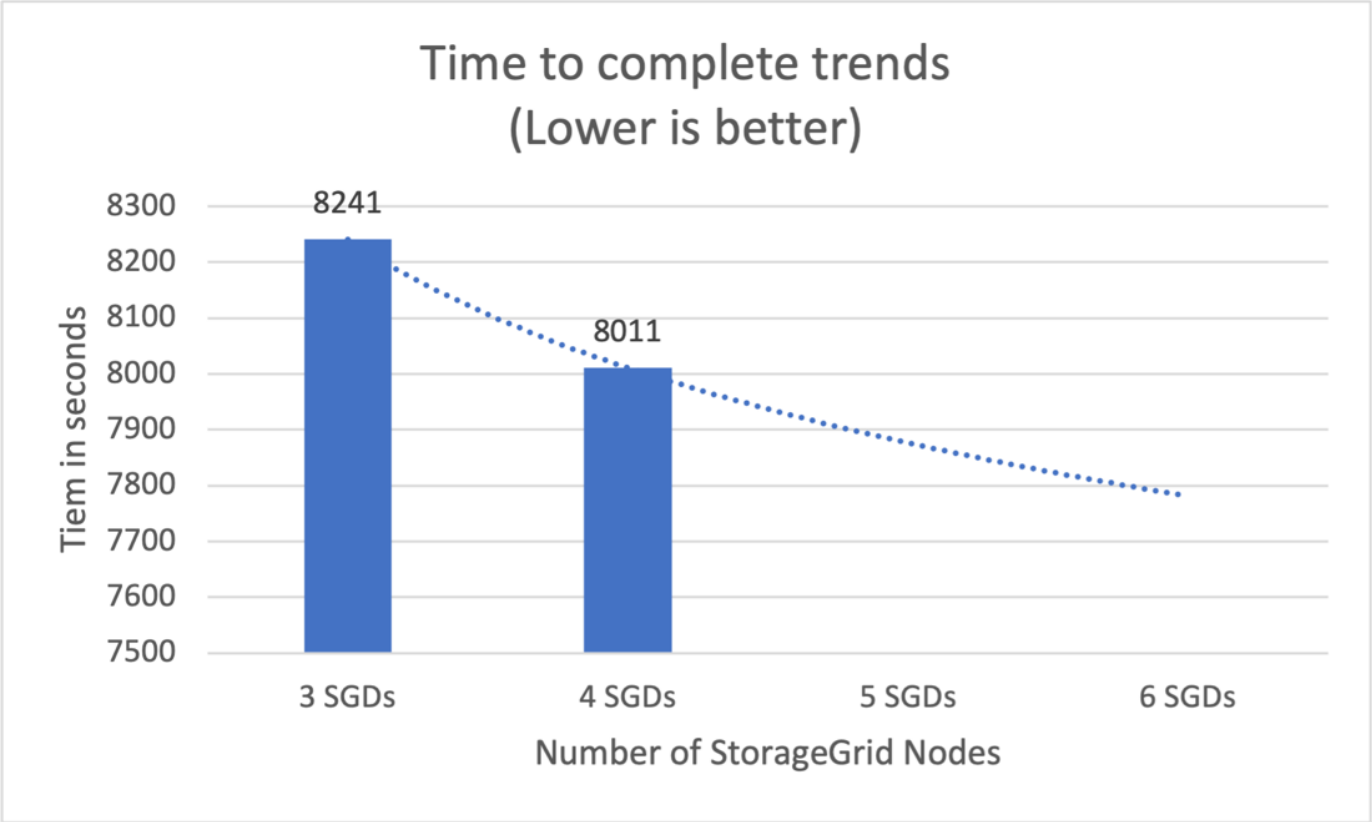
Questo test ha verificato le prestazioni di eliminazione di un archivio oggetti in presenza di un carico di lavoro di conservazione degli argomenti elevato. Il carico di lavoro di conservazione è stato generato utilizzando uno script di test che produce molti messaggi in parallelo a un argomento di test. L'argomento del test era la configurazione con un'impostazione di conservazione aggressiva basata sulle dimensioni e sul tempo, che causava la continua eliminazione del flusso di eventi dall'archivio oggetti. I segmenti vennero poi archiviati. Ciò ha portato a un gran numero di eliminazioni nell'archivio oggetti da parte del broker e alla raccolta delle prestazioni delle operazioni di eliminazione dell'archivio oggetti.

Test delle prestazioni con scalabilità

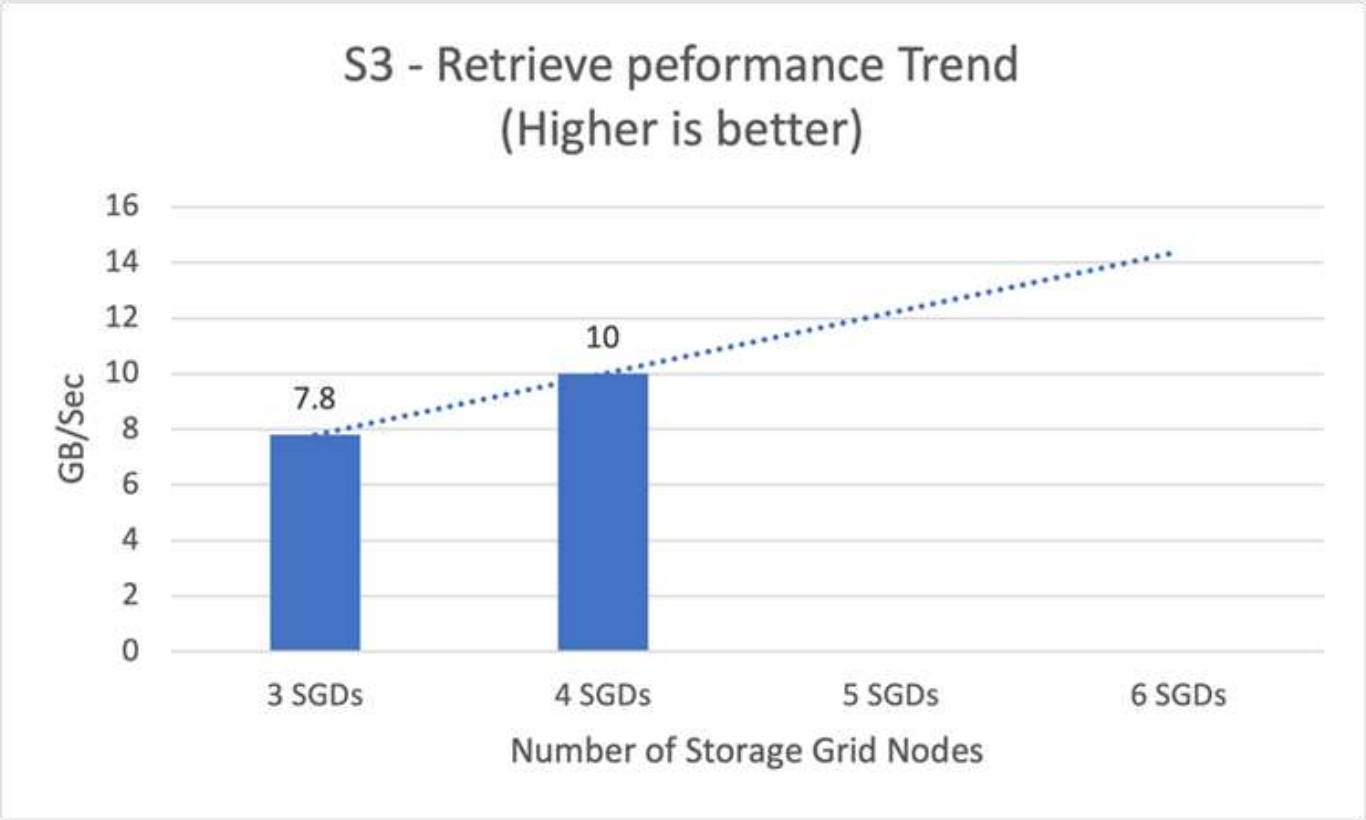
Abbiamo eseguito il test di archiviazione a livelli con tre o quattro nodi per i carichi di lavoro dei produttori e dei consumatori con la configurazione NetApp StorageGRID . Secondo i nostri test, il tempo di completamento e i risultati delle prestazioni sono stati

direttamente proporzionali al numero di nodi StorageGRID . La configurazione StorageGRID richiedeva un minimo di tre nodi.

- Il tempo necessario per completare le operazioni di produzione e consumo è diminuito linearmente all'aumentare del numero di nodi di stoccaggio.



- Le prestazioni dell'operazione di recupero s3 sono aumentate in modo lineare in base al numero di nodi StorageGRID . StorageGRID supporta fino a 200 nodi StorgeGRID.



Connettore confluente s3

Il connettore Amazon S3 Sink esporta i dati dagli argomenti Apache Kafka agli oggetti S3 nei formati Avro, JSON o Bytes. Il connettore sink Amazon S3 interroga periodicamente i dati da Kafka e a sua volta li carica su S3. Un partizionatore viene utilizzato per suddividere i dati di ogni partizione Kafka in blocchi. Ogni blocco di dati è rappresentato come un oggetto S3. Il nome della chiave codifica l'argomento, la partizione Kafka e l'offset iniziale di questo blocco di dati.

In questa configurazione, ti mostriamo come leggere e scrivere argomenti nell'archiviazione di oggetti da Kafka direttamente utilizzando il connettore sink Kafka s3. Per questo test abbiamo utilizzato un cluster Confluent autonomo, ma questa configurazione è applicabile anche a un cluster distribuito.

1. Scarica Confluent Kafka dal sito web di Confluent.
2. Decomprimi il pacchetto in una cartella sul tuo server.
3. Esporta due variabili.

```
Export CONFLUENT_HOME=/data/confluent/confluent-6.2.0
export PATH=$PATH:/data/confluent/confluent-6.2.0/bin
```

4. Per una configurazione Confluent Kafka autonoma, il cluster crea una cartella radice temporanea in `/tmp` Crea inoltre Zookeeper, Kafka, un registro di schemi, connect, un server ksql e cartelle del centro di controllo e copia i rispettivi file di configurazione da `$CONFLUENT_HOME` . Vedere il seguente esempio:

```
root@stlrx2540m1-108:~# ls -ltr /tmp/confluent.406980/
total 28
drwxr-xr-x 4 root root 4096 Oct 29 19:01 zookeeper
drwxr-xr-x 4 root root 4096 Oct 29 19:37 kafka
drwxr-xr-x 4 root root 4096 Oct 29 19:40 schema-registry
drwxr-xr-x 4 root root 4096 Oct 29 19:45 kafka-rest
drwxr-xr-x 4 root root 4096 Oct 29 19:47 connect
drwxr-xr-x 4 root root 4096 Oct 29 19:48 ksql-server
drwxr-xr-x 4 root root 4096 Oct 29 19:53 control-center
root@stlrx2540m1-108:~#
```

5. Configura Zookeeper. Se si utilizzano i parametri predefiniti non è necessario apportare alcuna modifica.

```
root@stlrx2540m1-108:~# cat
/tmp/confluent.406980/zookeeper/zookeeper.properties  | grep -iv ^#
dataDir=/tmp/confluent.406980/zookeeper/data
clientPort=2181
maxClientCnxns=0
admin.enableServer=false
tickTime=2000
initLimit=5
syncLimit=2
server.179=controlcenter:2888:3888
root@stlrx2540m1-108:~#
```

Nella configurazione di cui sopra, abbiamo aggiornato il `server. xxx` proprietà. Di default, per la selezione del leader Kafka sono necessari tre Zookeeper.

6. Abbiamo creato un file myid in `/tmp/confluent.406980/zookeeper/data` con un ID univoco:

```
root@stlrx2540m1-108:~# cat /tmp/confluent.406980/zookeeper/data/myid
179
root@stlrx2540m1-108:~#
```

Abbiamo utilizzato l'ultimo numero di indirizzi IP per il file myid. Abbiamo utilizzato valori predefiniti per le configurazioni Kafka, connect, control-center, Kafka, Kafka-rest, ksql-server e schema-registry.

7. Avviare i servizi Kafka.

```
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# confluent
local services  start
The local commands are intended for a single-node development
environment only,
NOT for production usage.

Using CONFLUENT_CURRENT: /tmp/confluent.406980
ZooKeeper is [UP]
Kafka is [UP]
Schema Registry is [UP]
Kafka REST is [UP]
Connect is [UP]
ksqlDB Server is [UP]
Control Center is [UP]
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

Per ogni configurazione è presente una cartella di registro, utile per risolvere i problemi. In alcuni casi, l'avvio dei servizi richiede più tempo. Assicurarsi che tutti i servizi siano attivi e funzionanti.

8. Installa Kafka Connect utilizzando `confluent-hub` .

```
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# ./confluent-
hub install confluentinc/kafka-connect-s3:latest
The component can be installed in any of the following Confluent
Platform installations:
  1. /data/confluent/confluent-6.2.0 (based on $CONFLUENT_HOME)
  2. /data/confluent/confluent-6.2.0 (where this tool is installed)
Choose one of these to continue the installation (1-2): 1
Do you want to install this into /data/confluent/confluent-
6.2.0/share/confluent-hub-components? (yN) y

Component's license:
Confluent Community License
http://www.confluent.io/confluent-community-license
I agree to the software license agreement (yN) y
Downloading component Kafka Connect S3 10.0.3, provided by Confluent,
Inc. from Confluent Hub and installing into /data/confluent/confluent-
6.2.0/share/confluent-hub-components
Do you want to uninstall existing version 10.0.3? (yN) y
Detected Worker's configs:
  1. Standard: /data/confluent/confluent-6.2.0/etc/kafka/connect-
distributed.properties
  2. Standard: /data/confluent/confluent-6.2.0/etc/kafka/connect-
standalone.properties
  3. Standard: /data/confluent/confluent-6.2.0/etc/schema-
```

```
registry/connect-avro-distributed.properties
  4. Standard: /data/confluent/confluent-6.2.0/etc/schema-
registry/connect-avro-standalone.properties
  5. Based on CONFLUENT_CURRENT:
/tmp/confluent.406980/connect/connect.properties
  6. Used by Connect process with PID 15904:
/tmp/confluent.406980/connect/connect.properties
Do you want to update all detected configs? (yN) y
Adding installation directory to plugin path in the following files:
  /data/confluent/confluent-6.2.0/etc/kafka/connect-
distributed.properties
  /data/confluent/confluent-6.2.0/etc/kafka/connect-
standalone.properties
  /data/confluent/confluent-6.2.0/etc/schema-registry/connect-avro-
distributed.properties
  /data/confluent/confluent-6.2.0/etc/schema-registry/connect-avro-
standalone.properties
  /tmp/confluent.406980/connect/connect.properties
  /tmp/confluent.406980/connect/connect.properties

Completed
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

È anche possibile installare una versione specifica utilizzando `confluent-hub install confluentinc/kafka-connect-s3:10.0.3`.

9. Per impostazione predefinita, `confluentinc-kafka-connect-s3` è installato in `/data/confluent/confluent-6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3`.
10. Aggiorna il percorso del plug-in con il nuovo `confluentinc-kafka-connect-s3`.

```
root@stlrx2540m1-108:~# cat /data/confluent/confluent-
6.2.0/etc/kafka/connect-distributed.properties | grep plugin.path
#
plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/co
nnectors,
plugin.path=/usr/share/java,/data/zookeeper/confluent/confluent-
6.2.0/share/confluent-hub-components,/data/confluent/confluent-
6.2.0/share/confluent-hub-components,/data/confluent/confluent-
6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3
root@stlrx2540m1-108:~#
```

11. Arrestare i servizi Confluent e riavviarli.

```
confluent local services  stop
confluent local services  start
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# confluent
local services  status
The local commands are intended for a single-node development
environment only,
NOT for production usage.

Using CONFLUENT_CURRENT: /tmp/confluent.406980
Connect is [UP]
Control Center is [UP]
Kafka is [UP]
Kafka REST is [UP]
ksqlDB Server is [UP]
Schema Registry is [UP]
ZooKeeper is [UP]
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

12. Configurare l'ID di accesso e la chiave segreta in `/root/.aws/credentials` file.

```
root@stlrx2540m1-108:~# cat /root/.aws/credentials
[default]
aws_access_key_id = xxxxxxxxxxxx
aws_secret_access_key = xxxxxxxxxxxxxxxxxxxxxxxxxx
root@stlrx2540m1-108:~#
```

13. Verificare che il bucket sia raggiungibile.

```
root@stlrx2540m4-01:~# aws s3 –endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 ls kafkasgdbucket1-2
2021-10-29 21:04:18       1388 1
2021-10-29 21:04:20       1388 2
2021-10-29 21:04:22       1388 3
root@stlrx2540m4-01:~#
```

14. Configurare il file delle proprietà s3-sink per la configurazione s3 e bucket.

```
root@stlrx2540m1-108:~# cat /data/confluent/confluent-
6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-
s3/etc/quickstart-s3.properties | grep -v ^#
name=s3-sink
connector.class=io.confluent.connect.s3.S3SinkConnector
tasks.max=1
topics=s3_testtopic
s3.region=us-west-2
s3.bucket.name=kafkasgdbucket1-2
store.url=http://kafkasgd.rtpppe.netapp.com:10444/
s3.part.size=5242880
flush.size=3
storage.class=io.confluent.connect.s3.storage.S3Storage
format.class=io.confluent.connect.s3.format.avro.AvroFormat
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartit
ioner
schema.compatibility=NONE
root@stlrx2540m1-108:~#
```

15. Importa alcuni record nel bucket s3.

```
kafka-avro-console-producer --broker-list localhost:9092 --topic
s3_topic \
--property
value.schema='{"type":"record","name":"myrecord","fields":[{"name":"f1",
"type":"string"}]}'
{"f1": "value1"}
{"f1": "value2"}
{"f1": "value3"}
{"f1": "value4"}
{"f1": "value5"}
{"f1": "value6"}
{"f1": "value7"}
{"f1": "value8"}
{"f1": "value9"}
```

16. Caricare il connettore s3-sink.

```
root@stlrx2540m1-108:~# confluent local services connect connector load
s3-sink  --config /data/confluent/confluent-6.2.0/share/confluent-hub-
components/confluentinc-kafka-connect-s3/etc/quickstart-s3.properties
The local commands are intended for a single-node development
environment only,
NOT for production usage.
https://docs.confluent.io/current/cli/index.html
{
  "name": "s3-sink",
  "config": {
    "connector.class": "io.confluent.connect.s3.S3SinkConnector",
    "flush.size": "3",
    "format.class": "io.confluent.connect.s3.format.avro.AvroFormat",
    "partitioner.class":
"io.confluent.connect.storage.partitioner.DefaultPartitioner",
    "s3.bucket.name": "kafkasgdbucket1-2",
    "s3.part.size": "5242880",
    "s3.region": "us-west-2",
    "schema.compatibility": "NONE",
    "storage.class": "io.confluent.connect.s3.storage.S3Storage",
    "store.url": "http://kafkasgd.rtpppe.netapp.com:10444/",
    "tasks.max": "1",
    "topics": "s3_testtopic",
    "name": "s3-sink"
  },
  "tasks": [],
  "type": "sink"
}
root@stlrx2540m1-108:~#
```

17. Controllare lo stato del sink s3.

```
root@stlrx2540m1-108:~# confluent local services connect connector
status s3-sink
The local commands are intended for a single-node development
environment only,
NOT for production usage.
https://docs.confluent.io/current/cli/index.html
{
  "name": "s3-sink",
  "connector": {
    "state": "RUNNING",
    "worker_id": "10.63.150.185:8083"
  },
  "tasks": [
    {
      "id": 0,
      "state": "RUNNING",
      "worker_id": "10.63.150.185:8083"
    }
  ],
  "type": "sink"
}
root@stlrx2540m1-108:~#
```

18. Controllare il registro per assicurarsi che s3-sink sia pronto ad accettare argomenti.

```
root@stlrx2540m1-108:~# confluent local services connect log
```

19. Consulta gli argomenti in Kafka.

```
kafka-topics --list --bootstrap-server localhost:9092
…
connect-configs
connect-offsets
connect-statuses
default_ksql_processing_log
s3_testtopic
s3_topic
s3_topic_new
root@stlrx2540m1-108:~#
```

20. Controllare gli oggetti nel bucket s3.

```
root@stlrx2540m1-108:~# aws s3 --endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 ls --recursive kafkasgdbucket1-
2/topics/
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000003.avro
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000006.avro
2021-10-29 21:24:08        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000009.avro
2021-10-29 21:24:08        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000012.avro
2021-10-29 21:24:09        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000015.avro
root@stlrx2540m1-108:~#
```

21. Per verificare il contenuto, copia ciascun file da S3 al tuo file system locale eseguendo il seguente comando:

```
root@stlrx2540m1-108:~# aws s3 --endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 cp s3://kafkasgdbucket1-
2/topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro
tes.avro
download: s3://kafkasgdbucket1-
2/topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro to
./tes.avro
root@stlrx2540m1-108:~#
```

22. Per stampare i record, utilizzare avro-tools-1.11.0.1.jar (disponibile nel "Archivi Apache").

```
root@stlrx2540m1-108:~# java -jar /usr/src/avro-tools-1.11.0.1.jar
tojson tes.avro
21/10/30 00:20:24 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
{"f1":"value1"}
{"f1":"value2"}
{"f1":"value3"}
root@stlrx2540m1-108:~#
```

Connettori Instaclustr Kafka Connect

Instaclustr supporta i connettori Kafka Connect e i loro dettagli - "Maggiori dettagli". Instaclustr fornisce connettori aggiuntivi "i loro dettagli"

Cluster autobilancianti confluenti

Se hai già gestito un cluster Kafka, probabilmente conosci le sfide che comporta la riassegnazione manuale delle partizioni a diversi broker per garantire che il carico di lavoro sia bilanciato nel cluster. Per le organizzazioni con grandi distribuzioni Kafka, riorganizzare grandi quantità di dati può essere scoraggiante, noioso e rischioso, soprattutto se le applicazioni mission-critical vengono sviluppate sul cluster. Tuttavia, anche per i casi d'uso più piccoli di Kafka, il processo richiede molto tempo ed è soggetto a errori umani.

Nel nostro laboratorio abbiamo testato la funzionalità di autobilanciamento dei cluster Confluent, che automatizza il ribilanciamento in base alle modifiche della topologia del cluster o al carico non uniforme. Il test di ribilanciamento Confluent aiuta a misurare il tempo necessario per aggiungere un nuovo broker quando un errore del nodo o il nodo di ridimensionamento richiedono il ribilanciamento dei dati tra i broker. Nelle configurazioni Kafka classiche, la quantità di dati da ribilanciare aumenta con la crescita del cluster, ma nell'archiviazione a livelli il ribilanciamento è limitato a una piccola quantità di dati. In base alla nostra convalida, il ribilanciamento nell'archiviazione a livelli richiede secondi o minuti in un'architettura Kafka classica e cresce in modo lineare con la crescita del cluster.

Nei cluster autobilancianti, i ribilanciamenti delle partizioni sono completamente automatizzati per ottimizzare la produttività di Kafka, accelerare il ridimensionamento del broker e ridurre l'onere operativo legato all'esecuzione di un cluster di grandi dimensioni. A regime, i cluster autobilancianti monitorano la distorsione dei dati tra i broker e riassegnano continuamente le partizioni per ottimizzare le prestazioni del cluster. Quando si aumenta o diminuisce la scalabilità della piattaforma, i cluster autobilancianti riconoscono automaticamente la presenza di nuovi broker o la rimozione di vecchi broker e attivano una successiva riassegnazione delle partizioni. Ciò consente di aggiungere e dismettere facilmente i broker, rendendo i cluster Kafka sostanzialmente più elastici. Questi vantaggi si ottengono senza bisogno di alcun intervento manuale, calcoli complessi o il rischio di errore umano che solitamente comportano le riassegnazioni delle partizioni. Di conseguenza, i ribilanciamenti dei dati vengono completati in molto meno tempo e puoi concentrarti su progetti di streaming di eventi di valore più elevato anziché dover supervisionare costantemente i tuoi cluster.

Instaclustr supporta anche funzionalità di auto-ribilanciamento ed è stato implementato per numerosi clienti.

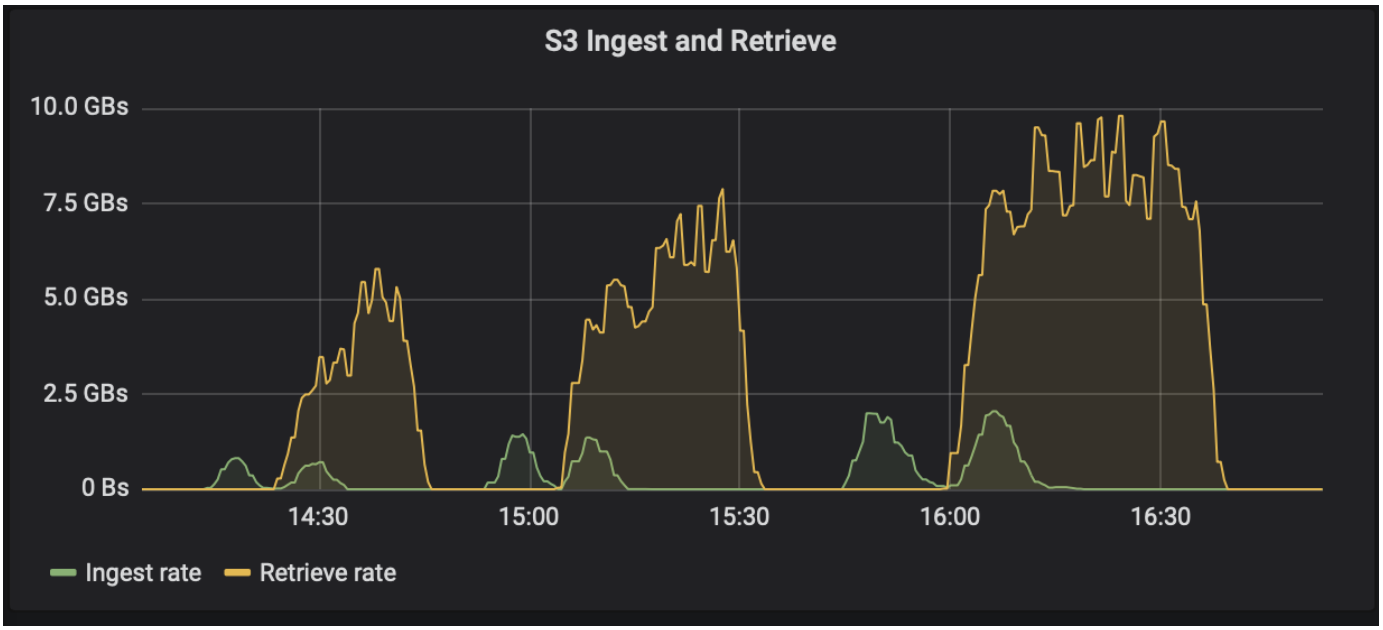
Linee guida per le migliori pratiche

Questa sezione presenta le lezioni apprese da questa certificazione.

- In base alla nostra convalida, l'archiviazione di oggetti S3 è la soluzione migliore per Confluent per conservare i dati.
- Possiamo utilizzare SAN ad alta capacità (in particolare FC) per mantenere i dati attivi del broker o il disco locale, perché, nella configurazione di archiviazione a livelli Confluent, la dimensione dei dati contenuti nella directory dei dati del broker si basa sulla dimensione del segmento e sul tempo di conservazione quando i dati vengono spostati nell'archiviazione degli oggetti.
- Gli archivi di oggetti offrono prestazioni migliori quando segment.bytes è più alto; abbiamo testato 512 MB.
- In Kafka, la lunghezza della chiave o del valore (in byte) per ogni record prodotto sull'argomento è

controllata da `length.key.value` parametro. Per StorageGRID, le prestazioni di acquisizione e recupero degli oggetti S3 sono aumentate a valori più elevati. Ad esempio, 512 byte hanno fornito un recupero di 5,8 GBps, 1024 byte hanno fornito un recupero s3 di 7,5 GBps e 2048 byte hanno fornito quasi 10 GBps.

La figura seguente presenta l'acquisizione e il recupero degli oggetti S3 in base a `length.key.value` .



- **Accordatura Kafka.** Per migliorare le prestazioni dell'archiviazione a livelli, è possibile aumentare TierFetcherNumThreads e TierArchiverNumThreads. Come linea guida generale, è consigliabile aumentare TierFetcherNumThreads in modo che corrisponda al numero di core fisici della CPU e aumentare TierArchiverNumThreads alla metà del numero di core della CPU. Ad esempio, nelle proprietà del server, se si dispone di una macchina con otto core fisici, impostare confluent.tier.fetcher.num.threads = 8 e confluent.tier.archiver.num.threads = 4.
- **Intervallo di tempo per l'eliminazione degli argomenti.** Quando un argomento viene eliminato, l'eliminazione dei file dei segmenti di registro nell'archiviazione degli oggetti non inizia immediatamente. Esiste invece un intervallo di tempo con un valore predefinito di 3 ore prima che i file vengano eliminati. È possibile modificare la configurazione confluent.tier.topic.delete.check.interval.ms per cambiare il valore di questo intervallo. Se elimini un argomento o un cluster, puoi anche eliminare manualmente gli oggetti nel rispettivo bucket.
- **ACL su argomenti interni di archiviazione a livelli.** Una best practice consigliata per le distribuzioni on-premise è quella di abilitare un'autorizzazione ACL sugli argomenti interni utilizzati per l'archiviazione a livelli. Impostare le regole ACL per limitare l'accesso a questi dati solo all'utente broker. In questo modo si proteggono gli argomenti interni e si impedisce l'accesso non autorizzato ai dati di archiviazione a livelli e ai metadati.

```
kafka-acls --bootstrap-server localhost:9092 --command-config adminclient-
configs.conf \
--add --allow-principal User:<kafka> --operation All --topic "_confluent-
tier-state"
```

Sostituisci l'utente `<kafka>` con il broker principale effettivo nella tua distribuzione.

Ad esempio, il comando `confluent-tier-state` imposta gli ACL sull'argomento interno per l'archiviazione a livelli. Attualmente esiste un solo argomento interno correlato allo storage a livelli. L'esempio crea un ACL che fornisce l'autorizzazione Kafka principale per tutte le operazioni sull'argomento interno.

Dimensionamento

Il dimensionamento Kafka può essere eseguito con quattro modalità di configurazione: semplice, granulare, inversa e partizioni.

Semplice

La modalità semplice è adatta ai nuovi utenti di Apache Kafka o ai casi d'uso iniziali. Per questa modalità, è necessario specificare requisiti quali velocità effettiva in MBps, fanout di lettura, conservazione e percentuale di utilizzo delle risorse (il 60% è l'impostazione predefinita). Si accede anche all'ambiente, ad esempio on-premise (bare-metal, VMware, Kubernetes o OpenStack) o cloud. Sulla base di queste informazioni, il dimensionamento di un cluster Kafka fornisce il numero di server necessari per il broker, lo zookeeper, i worker di connessione Apache Kafka, il registro degli schemi, un proxy REST, ksqlDB e il centro di controllo Confluent.

Per l'archiviazione a livelli, prendere in considerazione la modalità di configurazione granulare per il dimensionamento di un cluster Kafka. La modalità granulare è adatta agli utenti esperti di Apache Kafka o a casi d'uso ben definiti. Questa sezione descrive il dimensionamento per produttori, processori di flusso e consumatori.

Produttori

Per descrivere i produttori per Apache Kafka (ad esempio un client nativo, un proxy REST o un connettore Kafka), fornire le seguenti informazioni:

- **Nome.** Scintilla.
- **Tipo di produttore.** Applicazione o servizio, proxy (REST, MQTT, altro) e database esistente (RDBMS, NOSQL, altro). Puoi anche selezionare "Non lo so".
- **Capacità media.** In eventi al secondo (ad esempio 1.000.000).
- **Massima produttività.** In eventi al secondo (ad esempio 4.000.000).
- **Dimensione media del messaggio.** In byte, non compresso (max 1 MB; ad esempio 1000).
- **Formato del messaggio.** Le opzioni includono Avro, JSON, buffer di protocollo, binario, testo, "Non lo so" e altro.
- **Fattore di replicazione.** Le opzioni sono 1, 2, 3 (raccomandazione Confluent), 4, 5 o 6.
- **Tempo di conservazione.** Un giorno (ad esempio). Per quanto tempo desideri che i tuoi dati vengano archiviati in Apache Kafka? Inserire -1 con qualsiasi unità per un tempo infinito. La calcolatrice presuppone un tempo di conservazione di 10 anni per una conservazione infinita.
- Selezionare la casella di controllo "Abilitare l'archiviazione a livelli per ridurre il conteggio dei broker e consentire l'archiviazione infinita?"
- Quando è abilitato l'archiviazione a livelli, i campi di conservazione controllano il set di dati archiviati localmente sul broker. I campi di conservazione degli archivi controllano per quanto tempo i dati vengono conservati nell'archivio degli oggetti.
- **Conservazione in archivio.** Un anno (ad esempio). Per quanto tempo desideri che i tuoi dati rimangano archiviati? Inserisci -1 con qualsiasi unità per una durata infinita. Il calcolatore presuppone una

conservazione di 10 anni per una conservazione infinita.

- **Moltiplicatore di crescita.** 1 (ad esempio). Se il valore di questo parametro si basa sulla produttività attuale, impostarlo su 1. Per dimensionare in base alla crescita aggiuntiva, impostare questo parametro su un moltiplicatore di crescita.
- **Numero di istanze del produttore.** 10 (ad esempio). Quante istanze del produttore saranno in esecuzione? Questo input è necessario per incorporare il carico della CPU nel calcolo delle dimensioni. Un valore vuoto indica che il carico della CPU non viene incorporato nel calcolo.

Sulla base di questo esempio di input, il dimensionamento ha il seguente effetto sui produttori:

- Velocità media in byte non compressi: 1 GBps. Velocità di trasmissione massima in byte non compressi: 4 GBps. Velocità media in byte compressi: 400 MBps. Velocità di trasmissione massima in byte compressi: 1,6 GBps. Questo valore si basa su un tasso di compressione predefinito del 60% (è possibile modificarlo).
  - Spazio di archiviazione hotset totale on-broker richiesto: 31.104 TB, inclusa la replica, compresso. Spazio di archiviazione totale richiesto fuori dal broker: 378.432 TB, compresso. Utilizzo"https://fusion.netapp.com" per il dimensionamento StorageGRID .

Gli Stream Processor devono descrivere le loro applicazioni o servizi che consumano dati da Apache Kafka e li riproducono in Apache Kafka. Nella maggior parte dei casi sono creati in ksqlDB o Kafka Streams.

- **Nome.** Streamer scintillante.
- **Tempo di elaborazione.** Quanto tempo impiega questo processore per elaborare un singolo messaggio?
  - 1 ms (trasformazione semplice, senza stato) [esempio], 10 ms (operazione in memoria con stato).
  - 100 ms (operazione di rete o disco con stato), 1000 ms (chiamata REST di terze parti).
  - Ho confrontato questo parametro e so esattamente quanto tempo ci vuole.
- **Ritenzione dell'output.** 1 giorno (esempio). Un processore di flusso restituisce il suo output ad Apache Kafka. Per quanto tempo desideri che questi dati di output vengano archiviati in Apache Kafka? Inserisci -1 con qualsiasi unità per una durata infinita.
- Selezionare la casella di controllo "Abilitare l'archiviazione a livelli per ridurre il conteggio dei broker e consentire l'archiviazione infinita?"
- **Conservazione in archivio.** 1 anno (ad esempio). Per quanto tempo desideri che i tuoi dati rimangano archiviati? Inserisci -1 con qualsiasi unità per una durata infinita. Il calcolatore presuppone una conservazione di 10 anni per una conservazione infinita.
- **Percentuale di passaggio in uscita.** 100 (ad esempio). Un processore di flusso restituisce il suo output ad Apache Kafka. Quale percentuale del throughput in entrata verrà reinviata ad Apache Kafka? Ad esempio, se la velocità in ingresso è di 20 MBps e questo valore è 10, la velocità in uscita sarà di 2 MBps.
- Da quali applicazioni viene letto? Selezionare "Spark", il nome utilizzato nel dimensionamento basato sul tipo di produttore. Sulla base dei dati sopra riportati, è possibile aspettarsi i seguenti effetti del dimensionamento sulle istanze del processore di flusso e sulle stime delle partizioni degli argomenti:
- Questa applicazione di elaborazione di flussi richiede il seguente numero di istanze. Probabilmente anche gli argomenti in arrivo richiederanno questo numero di partizioni. Contattare Confluent per confermare questo parametro.
  - 1.000 per la produttività media senza moltiplicatore di crescita
  - 4.000 per la massima produttività senza moltiplicatore di crescita
  - 1.000 per la produttività media con un moltiplicatore di crescita
  - 4.000 per la massima produttività con un moltiplicatore di crescita

Consumatori

Descrivi le tue applicazioni o i tuoi servizi che consumano dati da Apache Kafka e non li restituiscono in Apache Kafka; ad esempio, un client nativo o Kafka Connector.

- **Nome.** Consumatore di scintille.
- **Tempo di elaborazione.** Quanto tempo impiega questo consumatore a elaborare un singolo messaggio?
  - 1 ms (ad esempio, un'attività semplice e senza stato come la registrazione)
  - 10 ms (scritture veloci su un datastore)
  - 100 ms (scritture lente su un datastore)
  - 1000 ms (chiamata REST di terze parti)
  - Qualche altro processo di riferimento di durata nota.
- **Tipo di consumatore.** Applicazione, proxy o sink per un datastore esistente (RDBMS, NoSQL, altro).
- Da quali applicazioni viene letto? Collegare questo parametro con le dimensioni del produttore e del flusso determinate in precedenza.

Sulla base dei dati di cui sopra, è necessario determinare le dimensioni per le istanze dei consumatori e le stime della partizione degli argomenti. Un'applicazione consumer richiede il seguente numero di istanze.

- 2.000 per la produttività media, nessun moltiplicatore di crescita
- 8.000 per la massima produttività, nessun moltiplicatore di crescita
- 2.000 per la produttività media, incluso il moltiplicatore di crescita
- 8.000 per la massima produttività, incluso il moltiplicatore di crescita

Probabilmente anche gli argomenti in arrivo necessitano di questo numero di partizioni. Contattare Confluent per conferma.

Oltre ai requisiti per produttori, processori di streaming e consumatori, è necessario fornire i seguenti requisiti aggiuntivi:

- **È tempo di ricostruire.** Ad esempio, 4 ore. Se un host broker Apache Kafka si guasta, i suoi dati vengono persi e viene predisposto un nuovo host per sostituire quello guasto, quanto velocemente deve ricostruirsi questo nuovo host? Lasciare vuoto questo parametro se il valore è sconosciuto.
- **Obiettivo di utilizzo delle risorse (percentuale).** Ad esempio, 60. Quanto vuoi che vengano utilizzati i tuoi host durante la velocità media? Confluent consiglia un utilizzo del 60%, a meno che non si utilizzino cluster autobilancianti Confluent, nel qual caso l'utilizzo può essere maggiore.

Descrivi il tuo ambiente

- **In quale ambiente verrà eseguito il tuo cluster?** Amazon Web Services, Microsoft Azure, Google Cloud Platform, bare-metal on premises, VMware on premises, OpenStack on premises o Kubernates on premises?
- **Dettagli dell'host.** Numero di core: 48 (ad esempio), tipo di scheda di rete (10GbE, 40GbE, 16GbE, 1GbE o altro tipo).
- **Volumi di archiviazione.** Host: 12 (ad esempio). Quanti dischi rigidi o SSD sono supportati per host? Confluent consiglia 12 dischi rigidi per host.
- **Capacità/volume di archiviazione (in GB).** 1000 (ad esempio). Quanto spazio di archiviazione può contenere un singolo volume in gigabyte? Confluent consiglia dischi da 1 TB.

- **Configurazione di archiviazione.** Come vengono configurati i volumi di archiviazione? Confluent consiglia RAID10 per sfruttare tutte le funzionalità di Confluent. Sono supportati anche JBOD, SAN, RAID 1, RAID 0, RAID 5 e altri tipi.
- **Capacità di trasmissione di un singolo volume (MBps).** 125 (ad esempio). Qual è la velocità in megabyte al secondo con cui un singolo volume di archiviazione può leggere o scrivere? Confluent consiglia dischi rigidi standard, che in genere hanno una velocità di trasmissione di 125 MBps.
- **Capacità di memoria (GB).** 64 (ad esempio).

Dopo aver determinato le variabili ambientali, seleziona Dimensiona il mio cluster. Sulla base dei parametri di esempio indicati sopra, abbiamo determinato le seguenti dimensioni per Confluent Kafka:

- **Apache Kafka.** Numero di broker: 22. Il cluster è vincolato allo storage. Valuta la possibilità di abilitare l'archiviazione a livelli per ridurre il numero di host e consentire uno spazio di archiviazione infinito.
- **Apache ZooKeeper.** Conteggio: 5; Apache Kafka Connect Workers: Conteggio: 2; Schema Registry: Conteggio: 2; REST Proxy: Conteggio: 2; ksqlDB: Conteggio: 2; Confluent Control Center: Conteggio: 1.

Utilizzare la modalità inversa per i team della piattaforma che non hanno in mente un caso d'uso. Utilizzare la modalità partizioni per calcolare quante partizioni richiede un singolo argomento. Vedere https://eventsizer.io per il dimensionamento basato sulle modalità inversa e partizioni.

Conclusione

Questo documento fornisce linee guida sulle best practice per l'utilizzo di Confluent Tiered Storage con storage NetApp , inclusi test di verifica, risultati delle prestazioni dello storage a livelli, ottimizzazione, connettori Confluent S3 e funzionalità di autobilanciamento. Considerando le policy ILM, le prestazioni di Confluent con più test delle prestazioni per la verifica e le API S3 standard del settore, l'archiviazione di oggetti NetApp StorageGRID è la scelta ottimale per l'archiviazione a livelli di Confluent.

Dove trovare ulteriori informazioni

Per saperne di più sulle informazioni descritte nel presente documento, consultare i seguenti documenti e/o siti web:

- Che cos'è Apache Kafka

  "https://www.confluent.io/what-is-apache-kafka/"

- Documentazione del prodotto NetApp

  "https://www.netapp.com/support-and-training/documentation/"

- Dettagli dei parametri del sink S3

  "https://docs.confluent.io/kafka-connect-s3-sink/current/configuration_options.html#s3-configuration-options"

- Apache Kafka

  "https://en.wikipedia.org/wiki/Apache_Kafka"

- Archiviazione infinita nella piattaforma Confluent

  "https://www.confluent.io/blog/infinite-kafka-storage-in-confluent-platform/"

- Confluent Tiered Storage: best practice e dimensionamento

  "https://docs.confluent.io/platform/current/kafka/tiered-storage.html#best-practices-and-recommendations"

- Connettore sink Amazon S3 per Confluent Platform

  "https://docs.confluent.io/kafka-connect-s3-sink/current/overview.html"

- Dimensionamento di Kafka

  "https://eventsizer.io"

- Dimensionamento StorageGRID

  "https://fusion.netapp.com/"

- Casi d'uso di Kafka

  "https://kafka.apache.org/uses"

- Cluster Kafka autobilancianti nella piattaforma confluente 6.0

  "https://www.confluent.io/blog/self-balancing-kafka-clusters-in-confluent-platform-6-0/"

  "https://www.confluent.io/blog/confluent-platform-6-0-delivers-the-most-powerful-event-streaming-platform-to-date/"

- Esempi di clienti Instaclustr e dettagli sui loro casi d'uso

  https://www.instaclustr.com/blog/netapp-and-pegasystems-open-source-support-package/, https://www.instaclustr.com/wp-content/uploads/Insta_Case_Study_Pegasystems_1_21sep25.pdf

  https://www.instaclustr.com/resources/customer-case-study-pubnub/

  https://www.instaclustr.com/resources/customer-case-study-tesouro/

Informazioni sul copyright

Informazioni sul marchio commerciale