Scania Engagement Report

Health Check recommendations

Rohit Shrivastava

2024-01-17

ABSTRACT

This document describes the recommendations provided by Confluent from the recent Professional Services engagement.

Discussions and Recommendations

Cost optimization

Scania's Kafka infrastructure is showing signs of underutilization, leading to potential cost inefficiencies. The analysis uncovered several key observations:

- **Low Data Throughput:** Average workload on the Kafka cluster is well below 2 MBps, with peak loads only reaching 7 MBps. This suggests the current infrastructure is significantly oversized for Scania's actual data flow.
- **Limited Disk Consumption:** Disk usage sits at an average of 160 GB per node, leaving considerable storage capacity untapped.
- **Subpar CPU and Memory Utilisation:** CPU utilisation across all nodes (including Kafka and other components) remains between 5-10%. Similarly, memory utilisation is minimal, indicating that current resources are not being fully leveraged.
- **Resource Fragmentation:** Each component runs on its own dedicated node pool, ensuring isolation but hindering optimal resource utilisation. This one-pod-per-node strategy currently requires 26 nodes in production solely for Confluent components.
- **Instance Type Mix:** The use of multiple instance types (m5.large, m5.xlarge, and m5.2xlarge) further adds to the complexity and potential for cost optimization.

Node pool

Based on Scania's low Kafka throughput, merging the current node pool presents a significant opportunity for cost optimization. Leveraging EKS' flexibility, we recommend consolidating into a single node pool for all Confluent components. This will ensure efficient resource utilisation and cost savings.

Considerations for Node Size:

- **Instance Type:** While arm-based instances offer potential cost benefits, their docker images are still in Early Access. Therefore, initially focusing on the current m5 series is more practical.
- **Rightsizing:** Opt for medium or large nodes (instead of smaller ones) to minimise overhead from kubelet and OS, while providing sufficient resources for pods.
- **Testing and Validation:** To validate the recommended configuration, we suggest testing the following options in a pre-production environment:
    - **Option 1**: 9 Nodes of m5.2xlarge (16vCPU, 32GB RAM) - Provides 72vCPU and 288GB RAM - Estimated 40% cost savings compared to current setup.
    - **Option 2**: 6 Nodes of m5.4xlarge (32vCPU, 64GB RAM) - Provides 96vCPU and 384GB RAM - Estimated 20% cost savings compared to current setup.

Additional Recommendations:

- **Auto Scaling:** Implement Cluster AutoScaler or Karpenter to dynamically manage node groups based on usage.
- **Pod Balancing:** Consider tools like Kubernetes Descheduler (GitHub - kubernetes-sigs/descheduler) to optimise pod distribution across nodes and prevent underutilization.
- **Continuous Monitoring:** Regularly monitor cluster usage and scale up manually if a static capacity model is chosen. Ensure no component suffers resource constraints.

Ensuring Optimal Resource Allocation for Confluent Pods in a Shared Node Pool

Key Considerations:

- **Shared Node Pool Implications:** When pods share nodes, setting appropriate resource requests and limits becomes crucial for efficient resource allocation and cluster stability.
- **Current Pod Configuration:** The lack of defined requests and limits in Scania's current setup, where each node runs a single pod, necessitates adjustments for a shared node pool scenario.

Recommendations for Resource Requests and Limits:

- **Kafka:** 16GB Memory Request, 16GB Memory Limit, 4 mCPU Request
- **Connect**: 6GB Memory Request, 16GB Memory Limit, 2 mCPU Request
- **Zookeeper**: 4GB Memory Request, 8GB Memory Limit, 1 mCPU Request
- **Control Center:** 4GB Memory Request, 8GB Memory Limit, 1 mCPU Request
- **Schema Registry:** 1GB Memory Request, 4GB Memory Limit, 0.5 mCPU Request
- **Rest Proxy:** 1GB Memory Request, 4GB Memory Limit, 0.5 mCPU Request
- **KsqlDB:** 1GB Memory Request, 2GB Memory Limit, 4 mCPU Request (Of course these are set to minimum and would depend on battery factory performance testing. Sizing requirements must come from them.)

Important note: Please note that the suggested memory limits of 16GB for Kafka (which can be extended up to 32GB/64GB depending on the instance type) are based on the current low throughput scenario and almost zero consumer lag. This may need to be adjusted upward as traffic increases.

Key Points:

- **Scheduling Guarantees:** K8s control plane schedules pods only on nodes meeting their resource requests, ensuring each pod receives its minimum required resources.

- **Flexibility and Optimization:** Not setting CPU limits allows pods to leverage available CPU capacity without hindering others, optimising overall resource usage.
- **Testing and Validation**: Pre-production testing is essential to confirm these configurations meet performance requirements and can handle traffic up to 10 MBps.
- **Official Kubernetes Documentation:** For in-depth understanding, refer to official Kubernetes documentation on resource management: Resource Management for Pods and Containers | Kubernetes

Additional Considerations:

- **Monitoring and Adjustment:** Continuously monitor resource utilisation and adjust requests/limits as needed to align with actual usage patterns.
- **Future Scaling:** Consider potential growth in traffic and plan for future resource requirements.

Enhancing Availability by Preventing Component Pod Colocation

Key Recommendation:

- **Leverage `oneReplicaPerNode` Setting:** Ensure this setting is configured to true within each Confluent For Kubernetes (CFK) resource. This crucial configuration prevents multiple pods of the same component from residing on the same node, mitigating potential single points of failure.

Rationale:

- **Component Isolation:** By distributing pods across different nodes, a node-level outage or failure will not compromise the availability of all replicas of a given component.
- **Resilience Enhancement:** This strategy safeguards against unexpected node-related issues, ensuring continuous operation of essential services.

Additional Considerations:

- **Monitoring and Alerting:** Establish robust monitoring and alerting mechanisms to promptly detect any colocation of pods that might occur due to configuration errors or unexpected events.
- **Node Failure Scenarios:** Regularly simulate node failures in testing environments to assess the cluster's resilience and validate the effectiveness of this strategy.

Optimising Replication for Enhanced Scalability and Availability

Current Deployment Topology:

- 3 Kafka Connect Clusters (aws, onprem, ME)
- 1 Replica for Rest Proxy
- 3 Replicas for Zookeeper

Recommendations:

- **Kafka Connect Clusters:**
    - **Consolidate `aws` and `onprem` clusters**: Since the `aws` cluster only runs one connector, it can be seamlessly migrated to the `onprem` cluster, resulting in efficient resource utilisation and simplified management. This reduces complexity and eliminates potential maintenance overhead for a rarely used cluster.
- **Rest Proxy:**
    - **Increase replicas to 2:** Although Rest Proxy usage is minimal, high availability remains crucial. Increasing replicas to 2 ensures service continuity even if one replica encounters issues, enhancing stability and user experience.
- **Zookeeper:**
    - **Consider increasing replicas to 5:** While the current 3 replicas can tolerate one node failure, 5 replicas improve fault tolerance and ensure the Zookeeper ensemble remains functional even if two nodes fail

simultaneously. This significantly strengthens cluster resilience for unforeseen events.

Leveraging Reduced Infrastructure Mode for Confluent Control Center

Scania can further optimise its Kafka infrastructure by utilising Confluent Control Center's Reduced Infrastructure Mode. This mode eliminates the monitoring features, transforming Control Center into a pure management tool, significantly reducing its resource footprint.

Benefits:

- **Reduced Resource Requirements:** Smaller memory and CPU requests/limits are appropriate for Control Center in Reduced Infrastructure mode, leading to cost savings and efficient resource utilisation.
- **Simplified Management:** The focus on pure management simplifies setup and maintenance, minimising operational overhead.
- **Enhanced Efficiency:** Eliminating unnecessary monitoring functionalities optimises resource allocation for core operational tasks.

Recommendations:

- **Configure Reduced Infrastructure Mode:** Activate this mode for Scania's Control Center instance to benefit from its resource-efficient design.
- **Adjust Resource Requests/Limits:** Update the suggested memory and CPU requirements (considering your previous recommendations) to reflect the reduced resource needs of Control Center in this mode.
- **Monitoring Integration:** For comprehensive observability, integrate external monitoring tools like Datadog or Prometheus to cover the functionality removed in Reduced Infrastructure mode. Create repeatable dashboards in each environment from code and promote to all environments for consistency.

Leveraging Tiered Storage for Cost-Effective Data Management

Unlocking Value with Tiered Storage:

Confluent's Tiered Storage feature holds immense potential for Scania to optimise storage costs and resource utilisation in its Kafka infrastructure. This feature seamlessly moves older data segments from expensive EBS volumes to cost-effective S3 buckets, while maintaining access for consumers.

Benefits for Scania:

- **Reduced Costs:** Pay only for storage actively used, significantly lowering costs associated with high-performance EBS volumes for infrequently accessed data.
- **Improved Infrastructure Efficiency:** Free up valuable EBS capacity for hot data, while keeping cold data readily available in S3.
- **Simplified Management:** Streamlined data lifecycle management through automated data movement based on configurable retention policies.

Implementation Considerations:

- **Topic-Specific Configuration:** Tiered Storage can be enabled on individual topics, allowing flexible management based on data access patterns. Scania can identify topics with and configure appropriate hotset thresholds (bytes or milliseconds) for automatic tiering.
- **Monitoring and Evaluation**: Closely monitor data access patterns and storage utilisation after enabling Tiered storage. Evaluate and adjust tier configurations as needed for optimal performance and cost efficiency.
- **Compacted Topic Compatibility**: Currently, Tiered Storage is not supported for compacted topics. If Scania utilises compacted topics, consider deferring tiered storage implementation for these specific topics until Confluent introduces compatibility in version 7.6 or later. Alternatively, analyse the feasibility of migrating compacted topics to non-compacted formats to leverage tiered storage benefits.

Next Steps:

- **Identify suitable topics:** Analyse access logs and usage patterns to identify candidate topics for tiered storage implementation.
- **Develop a configuration plan:** Define retention policies and hotset thresholds for each selected topic, prioritising frequently accessed data retention on EBS.
- **Testing and validation:** Perform thorough testing in a pre-production environment to ensure tiered storage functions as expected without impacting performance or availability.
- **Phased rollout:** Consider a phased rollout to gradually migrate data to S3, minimising potential disruption and allowing for continuous monitoring and refinement.

By strategically implementing Tiered Storage, Scania can achieve significant cost savings while maintaining performance and access to historical data. This cost-effective approach optimises resource utilisation and contributes to a more efficient and scalable Kafka infrastructure.

For more information refer to Tiered Storage | Confluent Documentation

Reduce the disk sizes

As of now disk usage is really low as compared to available storage. A separate document is shared with Scania to help reduce the EBS volumes.

Leveraging Ingress Controllers for Cost-Effective Traffic Management

Current Load Balancer Setup:

- Scania uses 4 sets of load balancers:
    - 2 sets for two client networks.
    - 2 additional sets for dual authentication (mTLS and OAuth) on each listener.
- This results in a significant cost due to the static charges per load balancer (24 in total).

Recommendation: Replacing Load Balancers with Ingress Controllers:

- **Reduced Cost:** Deploying an Ingress controller eliminates the dependency on numerous load balancers, leading to substantial cost savings. AWS charges for Ingress resources based on usage, not a static fee per instance.
- **High Availability**: Running the Ingress controller as a DaemonSet across nodes ensures high availability and fault tolerance. Traffic continues to flow even if individual nodes encounter issues.
- **Minimal Resource Utilisation**: Utilising a fraction of resources from each node for the Ingress controller minimises its impact on core Kafka operations.