

Capítulo 7

Administración de Hadoop

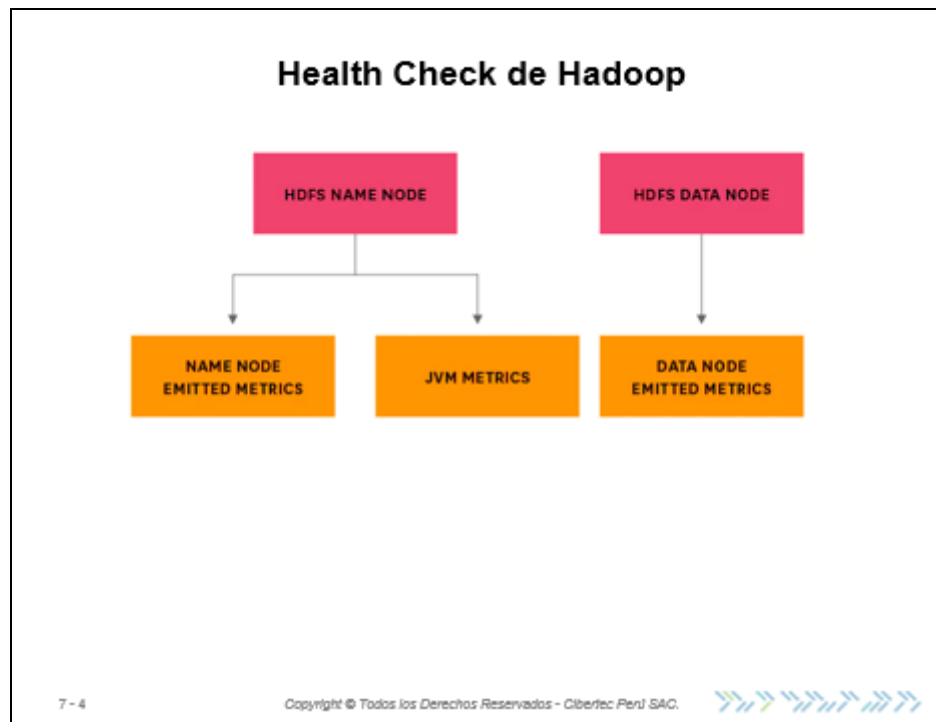
Al finalizar el capítulo, el alumno podrá:

- Comprenderá los parámetros de salud de un cluster Hadoop
- Implementará las buenas prácticas en la administración de Hadoop.

Temas

1. Health Check de Hadoop
2. Configuraciones y Parámetros

1. Health Check de Hadoop



Seguimiento orientado a servicios

Desde una perspectiva de operaciones, los clústeres de Hadoop son increíblemente elásticos frente a las fallas del sistema. Hadoop se diseñó teniendo en cuenta las fallas y puede tolerar el descenso de bastidores enteros.

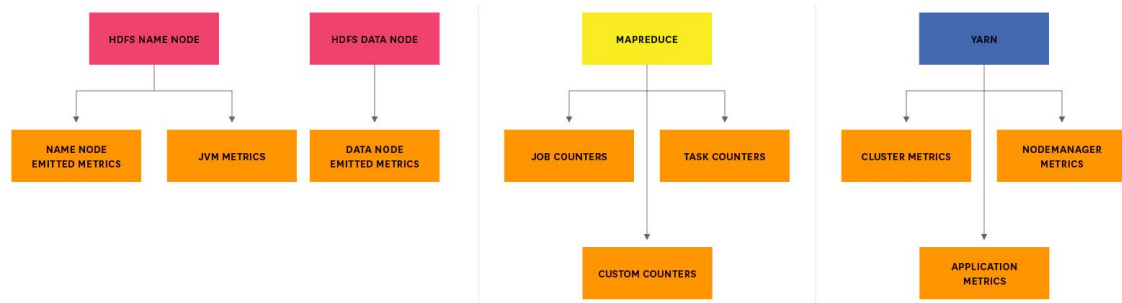
Monitorear Hadoop requiere una mentalidad diferente a la de monitorear algo como RabbitMQ-DataNodes y NodeManagers debe tratarse como ganado. Al igual que con los contenedores Docker, generalmente no le importa si falla un nodo de un solo trabajador, aunque la falla de un nodo maestro como uno de sus NameNodes requerirá una reparación bastante rápida. En términos generales, los indicadores de nivel de host individuales son menos importantes que los indicadores de nivel de servicio cuando se trata de Hadoop.

Métricas de rendimiento de Key Hadoop para monitorear

Cuando funciona correctamente, un clúster de Hadoop puede manejar una cantidad realmente enorme de datos: hay muchos clústeres de producción que manejan petabytes de datos cada uno. La supervisión de cada uno de los subcomponentes de Hadoop es esencial para mantener los trabajos en ejecución y el clúster.

Las métricas de Hadoop se pueden dividir en cuatro amplias categorías:

- Métricas de HDFS
- Contadores de MapReduce
- Métricas de YARN
- Métricas de ZooKeeper



Métricas de HDFS

Las métricas de HDFS se pueden descomponer aún más en dos categorías:

- Métricas de NameNode
- Métricas de DataNode

NameNode y DataNodes

Los DataNodes se comunican con su NameNode (y viceversa) a través de un latido periódico (por defecto cada tres segundos, establecido por `dfs.heartbeat.interval`). El latido contiene (entre otras cosas) información de salud sobre el DataNode. Es a través de este mecanismo que NameNode puede conocer el estado de todos los DataNodes en el clúster.

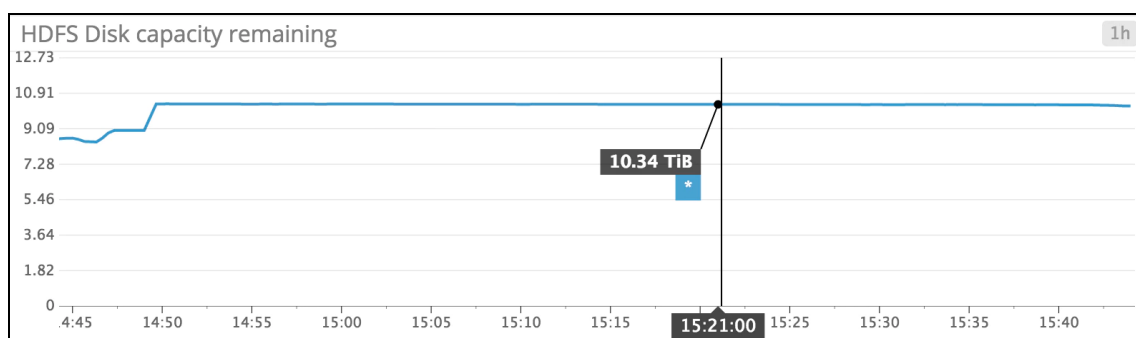
Si falla el NameNode sin que haya una copia en espera, los datos almacenados en el clúster dejarán de estar disponibles. Por lo tanto, monitorear el NameNode primario y su NameNode Secundario / Standby es crítico para asegurar una alta disponibilidad de clúster. Además de informar métricas sobre sí mismos, NameNodes también puede informar métricas sobre todos los DataNodes, proporcionando una buena vista panorámica del resto de su implementación de HDFS.

Las métricas de NameNode se pueden dividir en dos grupos:

- Métricas emitidas por NameNode
- Métricas de JVM de NameNode

Métricas emitidas por NameNode

Name	Description	<u>Metric Type</u>
CapacityRemaining	Available capacity	Resource: Utilization
CorruptBlocks/MissingBlocks	Number of corrupt/missing blocks	Resource: Error/Resource: Availability
VolumeFailuresTotal	Number of failed volumes	Resource: Error
NumLiveDataNodes/NumDeadDataNodes	Count of alive DataNodes/Count of dead DataNodes	Resource: Availability
FilesTotal	Total count of files tracked by the NameNode	Resource: Utilization
TotalLoad	Measure of file access across all DataNodes	Resource: Utilization
BlockCapacity/BlocksTotal	Maximum number of blocks allocable/Count of blocks tracked by NameNode	Resource: Utilization
UnderReplicatedBlocks	Count of under-replicated blocks	Resource: Availability
NumStaleDataNodes	Count of stale DataNodes	Resource: Availability



Métrica para alertar sobre: Capacidad Restante

CapacityRemaining representa la capacidad total disponible que queda en todo el clúster HDFS. Es una exageración decir que quedarse sin espacio en disco en HDFS causará BadThingsToHappen™. DataNodes que están fuera del espacio es probable que fallen al arrancar y, por supuesto, no se podrá escribir en DataNodes. Además, cualquier trabajo en ejecución que escriba datos temporales puede fallar debido a la falta de capacidad. Es una buena práctica asegurarse de que el uso del disco nunca supere el 80 por ciento de su capacidad.

Afortunadamente, HDFS facilita la creación de nuevos DataNodes para agregar capacidad (o puede agregar discos adicionales a los DataNodes existentes, por supuesto).

Métrica para alertar sobre: MissingBlocks

Los bloques dañados o faltantes pueden apuntar a un clúster no saludable. Normalmente, al recibir una solicitud de lectura, un DataNode ubicará el bloque solicitado y lo restituirá al cliente, junto con una suma de comprobación para los datos (que se calculó y almacenó en el DataNode cuando se escribió el bloque). Si la suma de comprobación para los datos recibidos no coincide con la suma de comprobación almacenada, el cliente informa la corrupción al NameNode y lee los datos de una de las otras réplicas. El NameNode, mientras tanto, programa una repetición del bloque de una de las copias sanas, y luego le dice al DataNode original que elimine su copia corrupta. Entonces, aunque es posible que vea bloques corruptos informados en su clúster de vez en cuando, eso no significa necesariamente que haya perdido datos.

Un bloque que falta es mucho peor que un bloque corrupto, porque un bloque que falta no se puede recuperar copiando una réplica. Un bloque faltante representa un bloque para el que no existe una copia conocida en el clúster. Eso no quiere decir que el bloque ya no exista; si una serie de DataNodes fuera desconectada para su mantenimiento, los bloques faltantes podrían ser informados hasta que se vuelvan a subir.

Puede que no le importe si tiene un bloque corrupto o dos en todo el clúster, pero si los bloques comienzan a faltar sin ningún mantenimiento programado, es motivo de preocupación.

Métrica para alertar sobre: VolumeFailuresTotal

En las versiones anteriores de Hadoop, la falla de un solo disco provocaría que un DataNode completo fuera desconectado. Afortunadamente, ya no es el caso: HDFS ahora permite que los discos fallen en su lugar, sin afectar las operaciones de DataNode, hasta que se alcanza un valor de umbral. (Esto se establece en cada DataNode a través de la propiedad `dfs.datanode.failed.volumes.tolerated`; por defecto es 0, lo que significa que cualquier falla de volumen cerrará el DataNode; en un clúster de producción donde los DataNodes suelen tener 6, 8 o 12) discos, establecer este parámetro en 1 o 2 suele ser la mejor práctica).

Aunque un volumen fallido no detendrá a su clúster, lo más probable es que desee saber cuándo se producen fallas de hardware, aunque solo sea para poder reemplazar el hardware que falló. Dado que HDFS se diseñó teniendo en cuenta las fallas de hardware, con una replicación predeterminada de tres, un volumen fallido nunca debe señalar la pérdida de datos.

Métrica para alertar sobre: NumDeadDataNodes

NumLiveDataNodes y NumDeadDataNodes juntos rastrean la cantidad de DataNodes en el clúster, por estado. Idealmente, su número de DataNodes en vivo será igual a la cantidad de DataNodes que haya aprovisionado para el clúster. Si este número cae inesperadamente, puede justificar una investigación.

NumDeadDataNodes, por otro lado, debe alertarse sobre. Cuando el NameNode no escucha desde un DataNode durante 30 segundos, ese DataNode se marca como "obsoleto". Si el DataNode no se comunica con el NameNode durante 10 minutos después de la transición al estado "obsoleto", el DataNode está marcado como "muerto". . "La muerte de un DataNode causa una ráfaga de actividad de red, ya que NameNode inicia la replicación de bloques perdidos en los nodos muertos.

Aunque la pérdida de un solo DataNode puede no afectar mucho el rendimiento, perder varios DataNodes comenzará a ser muy exigente con los recursos del clúster y podría provocar la pérdida de datos.

FilesTotal es un recuento de la cantidad de archivos que rastrea el NameNode. Debido a que NameNode almacena todos los metadatos en la memoria, cuanto mayor sea el número de archivos (o bloques, o directorios) rastreados, mayor será la memoria requerida por NameNode. La regla de oro para los requisitos de memoria de NameNode es que cada objeto (archivo, directorio, bloque) rastreado por NameNode consume aproximadamente 150 bytes de memoria. Si cada archivo se asigna a un bloque (los archivos a menudo abarcan varios bloques), eso es aproximadamente 300 bytes por archivo. Eso no parece mucho hasta que tenga en cuenta la replicación de datos: el factor de replicación predeterminado es 3 (por lo tanto, 900 bytes por archivo) y cada réplica agrega 16 bytes adicionales, para un total de ~ 1 KB de metadatos por archivo. Puede que todavía no suene como mucho, pero a medida que su clúster se expande para incluir millones de archivos, debe tener en cuenta este número y planear agregar memoria adicional a su NameNode.

TotalLoad es el número actual de acceso a archivos concurrentes (lectura / escritura) en todos los DataNodes. Esta métrica generalmente no es indicativa de un problema, aunque puede esclarecer problemas relacionados con la ejecución del trabajo. Debido a que los nodos de trabajador que ejecutan el daemon DataNode también realizan tareas de MapReduce, los períodos extendidos de alta I / O (indicado por TotalLoad alto) generalmente se traducen en un rendimiento de ejecución de trabajo degradado. El seguimiento de TotalLoad en el tiempo puede ayudar a llegar al fondo de los problemas de rendimiento laboral.

BlockCapacity / BlocksTotal: Like FilesTotal, controlar el número total de bloques en el clúster es esencial para la operación continua. La capacidad del bloque y la capacidad del archivo están estrechamente relacionadas, pero no son lo mismo (los archivos pueden abarcar múltiples bloques); monitorear tanto la capacidad de bloque como la de archivo es necesaria para garantizar un funcionamiento ininterrumpido. Si su clúster se acerca a los límites de la capacidad de bloque, puede agregar más fácilmente al abrir un nuevo nodo de datos y agregarlo al grupo, o agregar más discos a los nodos existentes. Una vez agregado, puede usar el equilibrador de Hadoop para equilibrar la distribución de bloques en los nodos de datos.

UnderReplicatedBlocks es la cantidad de bloques con replicación insuficiente. El factor de replicación de Hadoop es configurable por cliente o por archivo. El factor de replicación predeterminado es tres, lo que significa que cada bloque se almacenará en tres DataNodes. Si ve un gran y repentino aumento en el número de bloques poco replicados, es probable que un DataNode haya desaparecido; esto se puede verificar al correlacionar las métricas de bloque sub-replicadas con el estado de DataNodes (a través de NumLiveDataNodes y métricas relacionadas).

Imagen de DataNodes en vivo, inactiva y fuera de servicio

NumStaleDataNodes: cada tres segundos (de forma predeterminada), DataNodes envía un mensaje de latido al NameNode. Este latido funciona como un mecanismo de verificación de estado, pero también puede ser utilizado por NameNode para delegar el nuevo trabajo. Como se explicó anteriormente, los DataNodes que no envían un latido dentro de los 30 segundos se marcan como "obsoletos". Un DataNode podría no emitir un latido por varias razones: problemas de red o alta carga se encuentran entre las causas más comunes. A menos que haya desactivado las lecturas o escrituras en DataNodes obsoletos, definitivamente debe vigilar esta métrica. Un DataNode que no emitió un latido del corazón debido a una carga alta del sistema (o una red lenta) probablemente provocará retrasos en las lecturas o escrituras en ese DataNode. Sin embargo, en la práctica, un DataNode rancio probablemente esté muerto.

Incluso si está deshabilitado la lectura y la escritura en DataNodes obsoletos, debe supervisar los aumentos repentinos en la cantidad de DataNodes obsoletos, ya que en poco tiempo los nodos obsoletos pueden convertirse fácilmente en nodos muertos. Y los DataNodes obsoletos se escribirán independientemente de su configuración si la relación de DataNodes obsoletos a DataNodes totales excede el valor de `dfs.namenode.write.stale.datanode.ratio`

Métricas de JVM de NameNode

Debido a que NameNode se ejecuta en la Máquina Virtual Java (JVM), se basa en los procesos de recolección de basura de Java para liberar memoria. Cuanta más actividad en su clúster HDFS, más a menudo se ejecutará la recolección de basura.

Cualquiera que esté familiarizado con las aplicaciones Java sabe que la recolección de basura puede tener un alto costo de rendimiento. Consulte la documentación de Oracle para obtener una buena introducción a la recolección de basura de Java.

Si observa pausas excesivas durante la recolección de basura, puede considerar actualizar su versión JDK o el recolector de basura. Además, puede ajustar el tiempo de ejecución de Java para minimizar la recolección de basura.

Debido a que NameNode normalmente se aloja en una máquina con amplios recursos computacionales, se recomienda utilizar el recolector de elementos no utilizados CMS (ConcurrentMarkSweep), especialmente porque los cuellos de botella en NameNode darán como resultado un rendimiento degradado en todo el clúster. De la documentación de Oracle:

El compilador de CMS se debe usar para aplicaciones que requieren tiempos de pausa bajos y puede compartir recursos con el recolector de basura. Los ejemplos incluyen la aplicación de interfaz de usuario de escritorio que responde a eventos, un servidor web que responde a una solicitud o una base de datos que responde a consultas.

Las colecciones ConcurrentMarkSweep liberan la memoria no utilizada en la generación anterior del montón. CMS es una recolección de basura de baja pausa, lo que significa que, aunque detiene temporalmente los hilos de aplicación, lo hace de forma intermitente.

Para obtener más información sobre cómo optimizar la recolección de basura de NameNode, consulte esta excelente guía de Hortonworks.

Name	Description	Metric Type
ConcurrentMarkSweep count	Number of old-generation collections	Other
ConcurrentMarkSweep time	Elapsed time of old-generation collections, in milliseconds	Other

Si CMS tarda más de unos pocos segundos en completarse o está ocurriendo con mayor frecuencia, es posible que su NameNode no tenga suficiente memoria asignada a la JVM para funcionar de manera eficiente.

Métricas de DataNode

Las métricas de DataNode son métricas de nivel de host específicas para un DataNode en particular.

Name	Description	<u>Metric Type</u>
Remaining	Remaining disk space	Resource: Utilization
NumFailedVolumes	Number of failed storage volumes	Resource: Error

Remaining: como se menciona en la sección sobre métricas de NameNode, la falta de espacio en el disco puede causar una serie de problemas en un clúster. Si no se corrige, un solo DataNode que se quede sin espacio podría caer rápidamente en fallas en todo el clúster a medida que los datos se escriben en un grupo cada vez más reducido de DataNodes disponibles. El seguimiento de esta métrica a lo largo del tiempo es esencial para mantener un clúster saludable; Es posible que desee alertar sobre esta métrica cuando el espacio restante caiga peligrosamente bajo (menos del 10 por ciento).

NumFailedVolumes: de forma predeterminada, un solo volumen que falla en un DataNode hará que todo el nodo se desconecte. Dependiendo de su entorno, eso puede no ser deseable, especialmente dado que la mayoría de los DataNodes funcionan con múltiples discos. Si no ha configurado `dfs.datanode.failed.volumes.tolerated` en su configuración de HDFS, definitivamente debe realizar un seguimiento de esta métrica. Recuerde, cuando un DataNode se desconecta, el NameNode debe copiar cualquier bloque sub-replicado que se haya perdido en ese nodo, causando una explosión en el tráfico de red y una posible degradación del rendimiento. Para evitar una gran cantidad de actividad de red a partir de la falla de un solo disco (si su caso de uso lo permite), debe establecer el nivel tolerable de falla de volumen en su configuración de la siguiente manera:

```
<property>
```

```
    <name>dfs.datanode.failed.volumes.tolerated</name>
```

```
    <value>N</value>
```


</property>

donde N es la cantidad de fallas que un DataNode debe tolerar antes de apagarse.

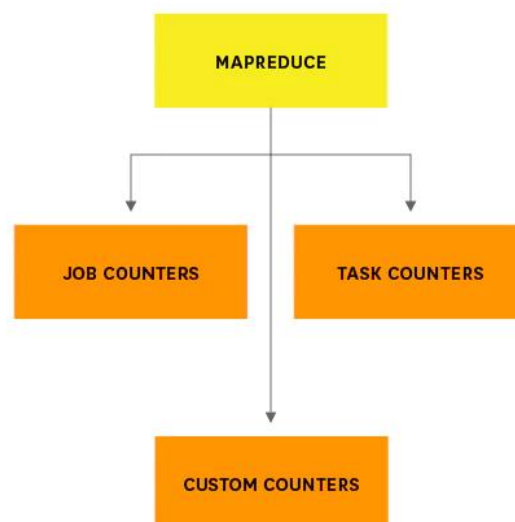
No olvide reiniciar su proceso de DataNode luego de un cambio de configuración.

La desventaja de habilitar esta característica es que, con menos discos para escribir, el rendimiento de un DataNode puede verse afectado. Establecer un valor razonable para las fallas toleradas variará según su caso de uso.

Contadores de MapReduce

El marco MapReduce expone una cantidad de contadores para rastrear estadísticas en la ejecución de trabajos de MapReduce. Los contadores son un mecanismo invaluable que le permite ver lo que realmente sucede durante una ejecución de trabajo de MapReduce. Proporcionan un recuento de la cantidad de veces que ocurrió un evento. Los contadores MapReduce se pueden dividir en cuatro categorías:

- job counters
- task counters
- custom counters
- file system counters



Los desarrolladores de aplicaciones de MapReduce pueden querer rastrear los siguientes contadores para encontrar cuellos de botella y posibles optimizaciones en sus aplicaciones.

Jobs Counters

Nombre del grupo: org.apache.hadoop.mapreduce.JobCounter

Counter Name	Description
MILLIS_MAPS/MILLIS_REDUCE	Processing time for maps/reduces
NUM_FAILED_MAPS/NUM_FAILED_REDUCE	Number of failed maps/reduces
RACK_LOCAL_MAPS/DATA_LOCAL_MAPS/OTHER_LOCAL_MAPS	Counters tracking where map tasks were executed

MILLIS_MAPS / MILLIS_REDUCE: estas métricas rastrean el tiempo de pared que se pasa en todo el mapa y reduce las tareas. Su utilidad completa se realiza cuando se realiza un seguimiento junto con el contador de tareas GC_TIME_MILLIS (ver a continuación); cuando se toma en conjunto, es posible determinar el porcentaje de tiempo dedicado a realizar la recolección de basura.

Usando los tres contadores, podemos calcular el porcentaje de tiempo pasado en la recolección de basura con la siguiente fórmula: $GC_TIME_PERCENTAGE = GC_TIME_MILLIS / (MILLIS_MAPS + MILLIS_REDUCE)$

Los trabajos en los que la recolección de basura consume un porcentaje significativo del tiempo de cálculo (por ejemplo, más del 20 por ciento) pueden beneficiarse al aumentar la cantidad de almacenamiento disponible para ellos.

NUM_FAILED_MAPS / NUM_FAILED_REDUCE: este par de indicadores realiza un seguimiento de la cantidad de tareas de mapa / reducción fallidas para un trabajo. Las tareas pueden fallar por muchas razones: la salida abrupta de la JVM, los errores de tiempo de ejecución y las tareas de suspensión se encuentran entre las más comunes. Las tareas fallidas se toleran, hasta el valor establecido por las propiedades `mapreduce.map.maxattempts` y `mapreduce.reduce.maxattempts` (establecido en cuatro, de forma predeterminada). Si una tarea falla `maxattempts` veces, se marca como fallida. Un trabajo fallará si fallan más de `mapreduce.map.failures.maxpercent` de las tareas de mapa en el trabajo o más de `mapreduce.reduce.failures.maxpercent` de las tareas de reducción.

Debido a que se esperan fallas, no debe alertar sobre esta métrica, aunque si ve que la misma tarea falla en varios DataNodes, puede valer la pena ingresar a los registros de trabajos o incluso al código de la aplicación para detectar errores. Si muchas tareas fallan en un DataNode en particular, el problema podría estar relacionado con el hardware.

DATA_LOCAL_MAPS / RACK_LOCAL_MAPS / OTHER_LOCAL_MAPS: estos contadores rastrean el número de tareas de mapa que se realizaron, agregadas por ubicación. Los datos se pueden ubicar: en el nodo que ejecuta la tarea de mapa, en un nodo en el mismo bastidor que el nodo que realiza la tarea de mapa o en un nodo ubicado en un bastidor diferente en otro

lugar del clúster. Recuerde de la parte 1 de esta serie que uno de los mayores puntos fuertes de Hadoop radica en su preferencia por mover los cálculos lo más cerca posible de los datos. Los beneficios de operar con datos locales incluyen una sobrecarga de red reducida y una ejecución más rápida. Estos beneficios se pierden cuando el cálculo debe realizarse en un nodo que no está físicamente cerca de los datos que se procesan.

Debido a que el envío de datos a través de la red lleva tiempo, la correlación del bajo rendimiento del trabajo con estos contadores puede ayudar a determinar por qué sufrió el rendimiento. Si se tenían que ejecutar muchos mapas en los nodos donde los datos no estaban alojados localmente, sería razonable esperar un rendimiento degradado.

Task Counters

Los contadores de tareas rastrean los resultados de las operaciones de tareas en conjunto. Cada contador a continuación representa el total en todas las tareas asociadas con un trabajo en particular.

Nombre del grupo: org.apache.hadoop.mapreduce.JobCounter

Counter Name	Description
REDUCE_INPUT_RECORDS	Number of input records for reduce tasks
SPIILLED_RECORDS	Number of records spilled to disk
GC_TIME_MILLIS	Processing time spent in garbage collection

REDUCE_INPUT_RECORDS: vigilar el número de registros de entrada para cada tarea de reducción en un trabajo es una de las mejores formas de identificar problemas de rendimiento relacionados con un conjunto de datos asimétrico. Los datos sesgados se caracterizan por una distribución desproporcionada de los registros de entrada en todos los reductores para un trabajo. Para dar un ejemplo concreto, considere la ejecución de la palabra canónica en un texto compuesto por 40,000 "the" y una "a". Si este texto se ejecutara a través de dos reductores, y cada uno obtuviera los valores asociados con una de las dos claves, el reductor que trabaja en el conjunto de datos más grande (el conjunto de "las" claves) tardaría mucho más en completarse que el reductor con una gran cantidad conjunto de datos más pequeño

Los conjuntos de datos sesgados pueden hacer que los trabajos tarden más en completarse de lo esperado. Desafortunadamente, los administradores de clústeres no pueden simplemente agregar más reductores para acelerar la ejecución; en casos como este, el trabajo de MapReduce probablemente se beneficie al agregar un paso de combinación.

SPIILLED_RECORDS: los datos de salida del mapa se almacenan en la memoria, pero cuando el búfer se llena los datos se vuelcan en el disco. Se engendra un hilo separado para fusionar todos los datos derramados en un único archivo clasificado más grande para los reductores. Los derrames en el disco deben minimizarse, ya que más de un derrame provocará un aumento en la actividad del disco ya que el hilo de fusión debe leer y escribir datos en el disco. El seguimiento de esta métrica puede ayudar a determinar si es necesario realizar cambios de configuración (como aumentar la memoria disponible para tareas de mapa) o si es necesario un procesamiento adicional (compresión) de los datos de entrada.

GC_TIME_MILLIS: Útil para determinar el porcentaje de tareas de tiempo dedicadas a la recolección de basura. Consulte la sección en `MILLIS_MAPS` / `MILLIS_REDUCES` para obtener más información.

Custom counters

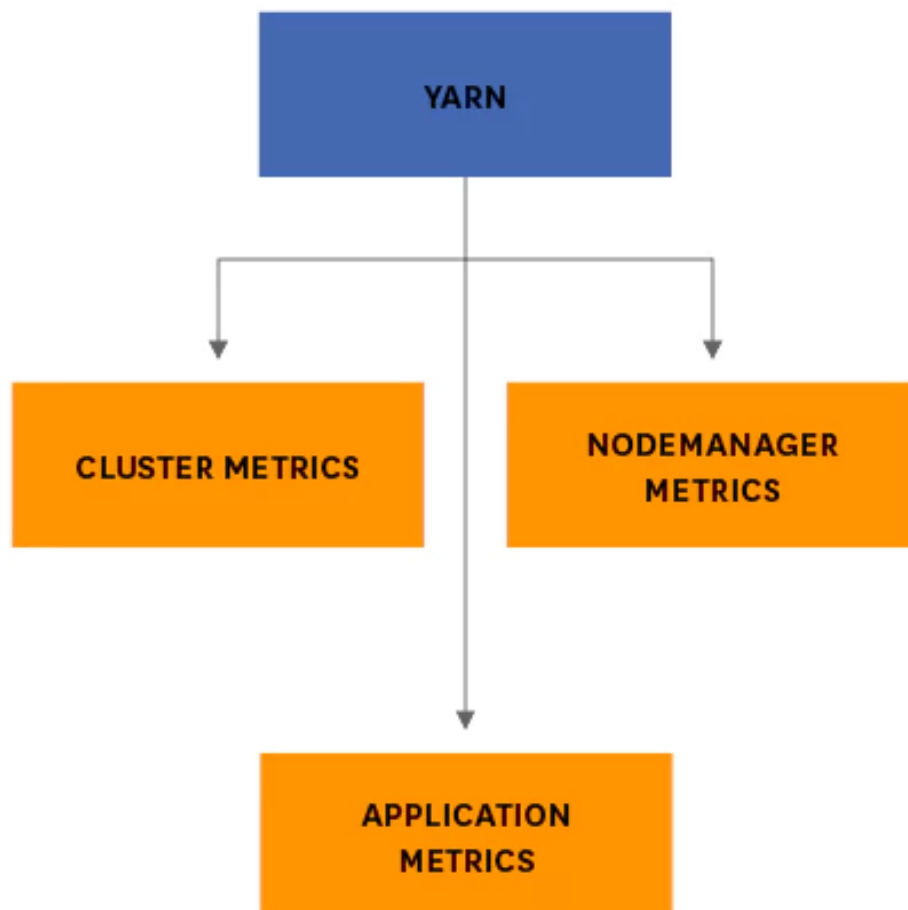
MapReduce permite a los usuarios implementar contadores personalizados que son específicos de su aplicación. Los contadores personalizados se pueden usar para rastrear recuentos más detallados, como contar el número de registros mal formados o faltantes.

Sin embargo, tenga en cuenta que, dado que todos los contadores residen en la memoria de la JVM de JobTracker, existe un límite práctico para la cantidad de contadores que debe usar, limitado por la cantidad de memoria física disponible. (Hadoop tiene un valor de configuración, `mapreduce.job.counters.max`, que limita el número de contadores personalizados que se pueden crear; de manera predeterminada, es 120).

Métricas de YARN

Las métricas de YARN se pueden agrupar en tres categorías distintas:

- Cluster metrics
- Application metrics
- NodeManager metrics



Métricas de Clusters

Las métricas de clúster proporcionan una vista de alto nivel de la ejecución de la aplicación YARN. Las métricas enumeradas aquí están agrupadas en todo el clúster.

Name	Description	<u>Metric Type</u>
unhealthyNodes	Number of unhealthy nodes	Resource: Error
activeNodes	Number of currently active nodes	Resource: Availability
lostNodes	Number of lost nodes	Resource: Error
appsFailed	Number of failed applications	Work: Error
totalMB/allocatedMB	Total amount of memory/amount of memory allocated	Resource: Utilization

Metric to alert on: unhealthyNodes

YARN considera que cualquier nodo con utilización de disco que exceda el valor especificado en la propiedad `yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage` (en `yarn-site.xml`) no es saludable. Un amplio espacio en disco es fundamental para garantizar el funcionamiento ininterrumpido de un clúster de Hadoop, y un gran número de nodos saludables (el número para alertar depende del tamaño de su clúster) debe investigarse y resolverse rápidamente. Una vez que un nodo está marcado como no saludable, otros nodos deben llenar el vacío, lo que puede provocar una cascada de alertas a medida que un nodo tras otro se acerca a la utilización total del disco.

Vale la pena mencionar que puede agregar sus propios controles de salud a YARN con un simple cambio de configuración. Pero tenga cuidado: si la secuencia de comandos falla por alguna razón, incluidos los permisos incorrectos, la ruta incorrecta, etc., hará que el nodo se marque como no saludable.

nodos activos / nodos perdidos: la métrica `activeNodes` rastrea el recuento actual de nodos activos, o normalmente en funcionamiento. Esta métrica debe permanecer estática en ausencia de mantenimiento anticipado. Si un `NodeManager` no puede mantener el contacto con `ResourceManager`, con el tiempo se marcará como "perdido" y sus recursos no estarán disponibles para la asignación. El umbral de tiempo de espera se puede configurar a través de la propiedad `yarn.resourcemanager.nm.liveness-monitor.expire-interval-ms` en `yarn-site.xml`, con un valor predeterminado de 10 minutos (600000 ms). Los nodos pueden perder contacto con `ResourceManager` por varias razones, incluidos problemas de red, fallas de hardware o un proceso bloqueado. Es bueno estar al tanto de los nodos perdidos, por lo que puede tomar medidas si el nodo nunca regresa.

`appsFailed`: suponiendo que esté ejecutando MapReduce en YARN, si el porcentaje de tareas fallidas de mapa o de reducción excede un umbral específico (configurado con `mapreduce.map.failures.maxpercent` y `mapreduce.reduce.failures.maxpercent`,

respectivamente), la aplicación como un todo fallar. Es importante tener en cuenta que YARN no puede distinguir entre un NodeManager fallido y una tarea pendiente: en cualquier caso, el NodeManager no informará al ResourceManager dentro del período de tiempo de espera y la tarea se volverá a ejecutar. Después de cuatro intentos infructuosos de ejecutar la tarea (configurados con `mapreduce.map.maxattempts` y `mapreduce.reduce.maxattempts`, respectivamente), se marcará como fallido.

En el caso de la falla de ResourceManager, el estado de la aplicación no se ve afectado en gran medida, aunque no se podrá enviar ningún trabajo nuevo y los trabajos en ejecución comenzarán a fallar. Debido a que los ApplicationMasters (en cada NodeManager) almacenan el estado de la aplicación (incluida la lista de tareas en ejecución / fallidas / canceladas), una vez que un nuevo ResourceManager se pone en línea (y suponiendo que el viejo estado guardado es recuperable), NodeManagers puede continuar donde lo dejaron apagado.

`totalMB` / `assignedMB`: cuando se toman en conjunto, las métricas `totalMB` y `assignMB` brindan una vista de alto nivel del uso de memoria de su clúster. La memoria es moneda en Hadoop, y si te estás acercando al techo de tu uso de memoria tienes un par de opciones. Para aumentar la memoria disponible para su clúster, puede agregar nuevos nodos NodeManager, modificar la cantidad de memoria reservada para las aplicaciones YARN (establecida por `yarn.nodemanager.resource.memory-mb` en `yarn-site.xml`, o cambiar la cantidad mínima) de RAM asignada a contenedores (establecida por la propiedad `yarn.scheduler.minimum-allocation-mb` en `yarn-site.xml`).

Tenga en cuenta que YARN puede sobrecomprometer recursos, lo que ocasionalmente puede traducirse en valores informados de `assignMB` que son más altos que el total de MB.

Application metrics

Las métricas de la aplicación proporcionan información detallada sobre la ejecución de aplicaciones individuales de YARN.

Name	Description	<u>Metric Type</u>
progress	Application execution progress meter	Work: Performance

Progress: Progress le brinda una ventana en tiempo real para la ejecución de una aplicación YARN. Su valor informado siempre estará en el rango de cero a uno (inclusive), con un valor de uno que indica la ejecución completa. Debido a que la ejecución de la aplicación muchas veces puede ser opaca cuando se ejecutan cientos de aplicaciones en miles de nodos, el seguimiento del progreso junto con otras métricas puede ayudarlo a determinar la causa de cualquier degradación del rendimiento. Deben investigarse las aplicaciones que abarcan períodos prolongados sin avanzar.

NodeManager metrics

Las métricas de NodeManager proporcionan información de recursos a nivel de nodo individual.

Name	Description	<u>Metric Type</u>
containersFailed	Number of containers that failed to launch	Resource: Error

containersFailed rastrea la cantidad de contenedores que no se pudieron iniciar en ese NodeManager en particular. Recuerde del primer artículo de la serie que los contenedores de NodeManager no son contenedores Docker.

Dejando a un lado las cuestiones de configuración, la causa más común de falla en el lanzamiento del contenedor está relacionada con el hardware. Si el disco de NodeManager está lleno, cualquier contenedor que inicie fallará. Del mismo modo, si el montón de NodeManager está configurado demasiado bajo, los contenedores no se abrirán. No es necesario que se le avise por cada contenedor que no pudo iniciarse, pero si un solo nodo tiene muchos contenedores fallidos, puede ser momento de investigar.

2. Configuraciones y Parámetros


Configuración y parámetros

Default

- core-default.xml
- hdfs-default.xml
- yarn-default.xml
- mapred-default.xml

Site

- /etc/hadoop/core-site.xml
- /etc/hadoop/hdfs-site.xml
- /etc/hadoop/yarn-site.xml
- /etc/hadoop/mapred-site.xml



7 - 9
Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.

Configuración del entorno de los daemons de Hadoop

Los administradores deben usar los scripts `/etc/hadoop/hadoop-env.sh` y, opcionalmente, los de `/etc/hadoop/mapred-env.sh` y `/etc/hadoop/yarn-env.sh` para realizar personalizaciones específicas del sitio del entorno de procesos de los daemons Hadoop.

Como mínimo, debe especificar `JAVA_HOME` para que esté correctamente definido en cada nodo remoto.

Los administradores pueden configurar demonios individuales usando las opciones de configuración que se muestran a continuación en la tabla:

Daemon	Environment Variable
NameNode	HDFS_NAMENODE_OPTS
DataNode	HDFS_DATANODE_OPTS
Secondary NameNode	HDFS_SECONDARYNAMENODE_OPTS
ResourceManager	YARN_RESOURCEMANAGER_OPTS
NodeManager	YARN_NODEMANAGER_OPTS
WebAppProxy	YARN_PROXYSERVER_OPTS
Map Reduce Job History Server	MAPRED_HISTORYSERVER_OPTS

Por ejemplo, para configurar Namenode para usar paraleloGC y un Heap Java de 4 GB, se debe agregar la siguiente declaración en `hadoop-env.sh`:

```
export HDFS_NAMENODE_OPTS="-XX:+UseParallelGC -Xmx4g"
```

Ver `/etc/hadoop/hadoop-env.sh` para otros ejemplos.

Otros parámetros de configuración útiles que puede personalizar incluyen:

- `HADOOP_PID_DIR` - El directorio donde se almacenan los archivos de Id. De proceso de los daemons.
- `HADOOP_LOG_DIR` - El directorio donde se almacenan los archivos de registro de los daemons. Los archivos de registro se crean automáticamente si no existen.
- `HADOOP_HEAPSIZE_MAX`: la cantidad máxima de memoria que se utilizará para el heapsize de Java. Las unidades compatibles con JVM también son compatibles aquí. Si no hay una unidad presente, se supondrá que el número está en megabytes. De forma predeterminada, Hadoop permitirá que la JVM determine cuánto usar. Este valor puede anularse por daemon utilizando la variable `_OPTS` adecuada enumerada anteriormente. Por ejemplo, establecer `HADOOP_HEAPSIZE_MAX = 1g` y `HADOOP_NAMENODE_OPTS = "-Xmx5g"` configurará el NameNode con 5GB de almacenamiento dinámico.
- En la mayoría de los casos, debe especificar los directorios `HADOOP_PID_DIR` y `HADOOP_LOG_DIR` de modo que solo puedan escribirlos los usuarios que ejecutarán los daemons de hadoop. De lo contrario, existe la posibilidad de un ataque de enlace simbólico.

También es tradicional configurar `HADOOP_HOME` en la configuración del entorno de shell de todo el sistema. Por ejemplo, un script simple dentro de `/etc/profile.d`:

```
HADOOP_HOME=/path/to/hadoop

export HADOOP_HOME
```

Configurando los Daemons de Hadoop

Esta sección trata sobre los parámetros importantes que deben especificarse en los archivos de configuración dados:

`/etc/hadoop/core-site.xml`

Parameter	Value	Notes
<code>fs.defaultFS</code>	NameNode URI	hdfs://host:port/
<code>io.file.buffer.size</code>	131072	Size of read/write buffer used in SequenceFiles.

`/etc/hadoop/hdfs-site.xml`

Configuraciones para NameNode:

Parameter	Value	Notes
dfs.namenode.name.dir	Path on the local filesystem where the NameNode stores the namespace and transactions logs persistently.	If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.
dfs.hosts / dfs.hosts.exclude	List of permitted/excluded DataNodes.	If necessary, use these files to control the list of allowable datanodes.
dfs.blocksize	268435456	HDFS blocksize of 256MB for large file-systems.
dfs.namenode.handler.count	100	More NameNode server threads to handle RPCs from large number of DataNodes.

Configuraciones para DataNode:

Parameter	Value	Notes
dfs.datanode.data.dir	Comma separated list of paths on the local filesystem of a DataNode where it should store its blocks.	If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices.

/etc/hadoop/yarn-site.xml**Configuraciones para ResourceManager y NodeManager:**

Parameter	Value	Notes
yarn.acl.enable	true / false	Enable ACLs? Defaults to <i>false</i> .
yarn.admin.acl	Admin ACL	ACL to set admins on the cluster. ACLs are of for <i>comma-separated-users space comma-separated-groups</i> . Defaults to special value of <i>*</i> which means <i>anyone</i> . Special value of just <i>space</i> means no one has access.
yarn.log-aggregation-enable	false	Configuration to enable or disable log aggregation

Configuraciones para ResourceManager:

Parameter	Value
yarn.resourcemanager.address	ResourceManagerhost:port for clients to submit jobs.
yarn.resourcemanager.scheduler.address	ResourceManagerhost:port for ApplicationMasters to talk to Scheduler to obtain resources.
yarn.resourcemanager.resource-tracker.address	ResourceManagerhost:port for NodeManagers.
yarn.resourcemanager.admin.address	ResourceManagerhost:port for administrative commands.
yarn.resourcemanager.webapp.address	ResourceManagerweb-ui host:port.

Parameter	Value
yarn.resourcemanager.hostname	ResourceManagerhost.
yarn.resourcemanager.scheduler.class	ResourceManagerScheduler class.
yarn.scheduler.minimum-allocation-mb	Minimum limit of memory to allocate to each container request at the Resource Manager.
yarn.scheduler.maximum-allocation-mb	Maximum limit of memory to allocate to each container request at the Resource Manager.
yarn.resourcemanager.nodes.include-path / yarn.resourcemanager.nodes.exclude-path	List of permitted/excluded NodeManagers.

Configuraciones para NodeManager:

Parameter	Value
yarn.nodemanager.resource.memory-mb	Resource i.e. available physical memory, in MB, for given NodeManager
yarn.nodemanager.vmem-pmem-ratio	Maximum ratio by which virtual memory usage of tasks may exceed physical memory
yarn.nodemanager.local-dirs	Comma-separated list of paths on the local filesystem where intermediate data is written.
yarn.nodemanager.log-dirs	Comma-separated list of paths on the local filesystem where logs are written.
yarn.nodemanager.log.retain-seconds	10800
yarn.nodemanager.remote-app-log-dir	/logs
yarn.nodemanager.remote-app-log-dir-suffix	logs
yarn.nodemanager.aux-services	mapreduce_shuffle
yarn.nodemanager.env-whitelist	Environment properties to be inherited by containers from NodeManagers

Configuraciones para el servidor de historial (necesita ser movido a otro lugar):

Parameter	Value
yarn.log-aggregation.retain-seconds	-1
yarn.log-aggregation.retain-check-interval-seconds	-1

/etc/hadoop/mapred-site.xml

Configuraciones para aplicaciones de MapReduce:

Parameter	Value
mapreduce.framework.name	yarn
mapreduce.map.memory.mb	1536
mapreduce.map.java.opts	-Xmx1024M
mapreduce.reduce.memory.mb	3072
mapreduce.reduce.java.opts	-Xmx2560M
mapreduce.task.io.sort.mb	512
mapreduce.task.io.sort.factor	100

arameter	Value
mapreduce.reduce.shuffle.parallelcopies	50

Configuraciones para MapReduce JobHistory Server:

Parameter	Value
mapreduce.jobhistory.address	MapReduce JobHistory Server <i>host:port</i>
mapreduce.jobhistory.webapp.address	MapReduce JobHistory Server Web UI <i>host:port</i>
mapreduce.jobhistory.intermediate-done-dir	/mr-history/tmp
mapreduce.jobhistory.done-dir	/mr-history/done

Bibliografía

“Configuring Environment of Hadoop Daemons”

http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html#Configuring_Hadoop_in_Non-Secure_Mode

Consulta: 01 de diciembre del 2017

“How to monitor Hadoop metrics”

<https://www.datadoghq.com/blog/monitor-hadoop-metrics/>

Consulta: 01 de diciembre del 2017