HDFS-Architektur

Einführung

Das Hadoop Distributed File System (HDFS) ist ein verteiltes Dateisystem, das für die Ausführung auf Standardhardware entwickelt wurde. Es hat viele Ähnlichkeiten mit vorhandenen verteilten Dateisystemen. Die Unterschiede zu anderen verteilten Dateisystemen sind jedoch erheblich. HDFS ist sehr fehlertolerant und für die Bereitstellung auf kostengünstiger Hardware ausgelegt. HDFS bietet Zugriff auf Anwendungsdaten mit hohem Durchsatz und eignet sich für Anwendungen mit großen Datenmengen. HDFS lockert einige POSIX-Anforderungen, um den Streaming-Zugriff auf Dateisystemdaten zu ermöglichen. HDFS wurde ursprünglich als Infrastruktur für das Web-Suchmaschinenprojekt Apache Nutch entwickelt. HDFS ist Teil des Apache Hadoop Core-Projekts. Die Projekt-URL lautet http://hadoop.apache.org/

.

Annahmen und Ziele

Hardwarefehler

Hardwarefehler sind eher die Norm als die Ausnahme. Eine HDFS-Instanz kann aus Hunderten oder Tausenden von Servercomputern bestehen, auf denen jeweils ein Teil der Daten des Dateisystems gespeichert ist. Die Tatsache, dass es eine große Anzahl von Komponenten gibt und dass jede Komponente eine nicht triviale Ausfallwahrscheinlichkeit aufweist, bedeutet, dass einige Komponenten von HDFS immer nicht funktionsfähig sind. Daher ist die Erkennung von Fehlern und deren schnelle, automatische Behebung ein zentrales architektonisches Ziel von HDFS.

Streaming-Datenzugriff

Anwendungen, die unter HDFS ausgeführt werden, benötigen Streaming-Zugriff auf ihre Datensätze. Es handelt sich nicht um Allzweckanwendungen, die normalerweise auf Allzweckdateisystemen ausgeführt werden. HDFS ist eher für die Stapelverarbeitung als für die interaktive Verwendung durch Benutzer konzipiert. Der Schwerpunkt liegt eher auf einem hohen Durchsatz des Datenzugriffs als auf einer geringen Latenz des Datenzugriffs. POSIX stellt viele harte Anforderungen, die für Anwendungen, die auf HDFS ausgerichtet sind, nicht benötigt werden. Die POSIX-Semantik in einigen Schlüsselbereichen wurde gehandelt, um die Datendurchsatzraten zu erhöhen.

Große Datenmengen

Anwendungen, die unter HDFS ausgeführt werden, verfügen über große Datenmengen. Eine typische Datei in HDFS hat eine Größe von Gigabyte bis Terabyte. Daher ist HDFS so abgestimmt, dass es große Dateien unterstützt. Es sollte eine hohe aggregierte Datenbandbreite bieten und auf Hunderte von Knoten in einem einzelnen Cluster skaliert werden können. Es sollte zig Millionen Dateien in einer einzigen Instanz unterstützen.

Einfaches Kohärenzmodell

HDFS-Anwendungen benötigen ein Zugriffsmodell für Dateien, bei dem einmal geschrieben und gelesen werden muss. Eine einmal erstellte, geschriebene und geschlossene Datei muss nur zum Anhängen und Abschneiden geändert werden. Das Anhängen des Inhalts an das Ende der Dateien wird unterstützt, kann jedoch nicht an einer beliebigen Stelle aktualisiert werden. Diese Annahme vereinfacht Datenkohärenzprobleme und ermöglicht den Datenzugriff mit hohem Durchsatz. Eine MapReduce-Anwendung oder eine Webcrawler-Anwendung passt perfekt zu diesem Modell.

"Verschieben von Berechnungen ist billiger als Verschieben von Daten"

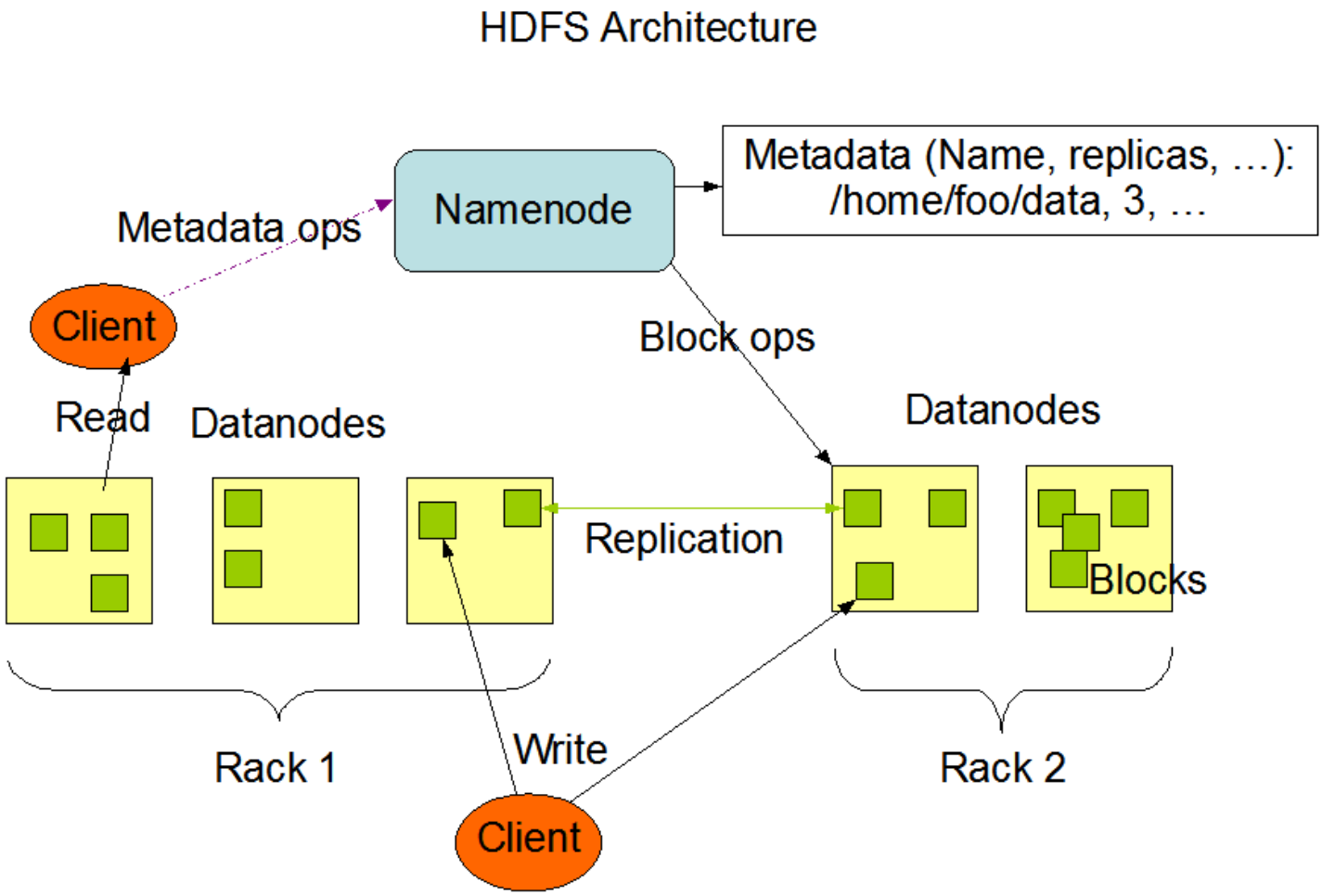
Eine von einer Anwendung angeforderte Berechnung ist viel effizienter, wenn sie in der Nähe der Daten ausgeführt wird, mit denen sie arbeitet. Dies gilt insbesondere dann, wenn die Größe des Datensatzes sehr groß ist. Dies minimiert die Überlastung des Netzwerks und erhöht den Gesamtdurchsatz des Systems. Die Annahme ist, dass es häufig besser ist, die Berechnung näher an den Ort zu migrieren, an dem sich die Daten befinden, als die Daten an den Ort zu verschieben, an dem die Anwendung ausgeführt wird. HDFS bietet Schnittstellen für Anwendungen, mit denen sie sich näher an den Ort der Daten bewegen können.

Portabilität auf heterogenen Hardware- und Softwareplattformen

HDFS wurde so konzipiert, dass es problemlos von einer Plattform auf eine andere übertragen werden kann. Dies erleichtert die breite Akzeptanz von HDFS als Plattform der Wahl für eine große Anzahl von Anwendungen.

NameNode und DataNodes

HDFS verfügt über eine Master / Slave-Architektur. Ein HDFS-Cluster besteht aus einem einzelnen NameNode, einem Master-Server, der den Dateisystem-Namespace verwaltet und den Zugriff von Clients auf Dateien regelt. Darüber hinaus gibt es eine Reihe von DataNodes, normalerweise einen pro Knoten im Cluster, die den Speicher verwalten, der an die Knoten angeschlossen ist, auf denen sie ausgeführt werden. HDFS macht einen Dateisystem-Namespace verfügbar und ermöglicht das Speichern von Benutzerdaten in Dateien. Intern wird eine Datei in einen oder mehrere Blöcke aufgeteilt und diese Blöcke werden in einem Satz von DataNodes gespeichert. Der NameNode führt Dateisystem-Namespace-Vorgänge wie das Öffnen, Schließen und Umbenennen von Dateien und Verzeichnissen aus. Außerdem wird die Zuordnung von Blöcken zu DataNodes festgelegt. Die DataNodes sind für die Bearbeitung von Lese- und Schreibanforderungen von den Clients des Dateisystems verantwortlich. Die DataNodes führen auch das Erstellen, Löschen von Blöcken durch.



Der NameNode und der DataNode sind Softwareteile, die für die Ausführung auf Standardmaschinen entwickelt wurden. Auf diesen Computern wird normalerweise ein GNU / Linux-Betriebssystem ausgeführt. HDFS wird in der Java-Sprache erstellt. Auf jedem Computer, der Java unterstützt, kann der NameNode oder die DataNode-Software ausgeführt werden. Durch die Verwendung der hoch portablen Java-Sprache kann HDFS auf einer Vielzahl von Computern bereitgestellt werden. Eine typische Bereitstellung verfügt über einen dedizierten Computer, auf dem nur die NameNode-Software ausgeführt wird. Auf jedem der anderen Computer im Cluster wird eine Instanz der DataNode-Software ausgeführt. Die Architektur schließt nicht aus, dass mehrere DataNodes auf demselben Computer ausgeführt werden, sondern in einer realen Bereitstellung, die selten der Fall ist.

Das Vorhandensein eines einzelnen NameNode in einem Cluster vereinfacht die Architektur des Systems erheblich. Der NameNode ist der Schiedsrichter und das Repository für alle HDFS-Metadaten. Das System ist so konzipiert, dass Benutzerdaten niemals durch den NameNode fließen.

Der Dateisystem-Namespace

HDFS unterstützt eine traditionelle hierarchische Dateiorganisation. Ein Benutzer oder eine Anwendung kann Verzeichnisse erstellen und Dateien in diesen Verzeichnissen speichern. Die Dateisystem-Namespace-Hierarchie ähnelt den meisten anderen vorhandenen Dateisystemen. Man kann Dateien erstellen und entfernen, eine Datei von einem Verzeichnis in ein anderes verschieben oder eine Datei umbenennen. HDFS unterstützt Benutzerkontingente und Zugriffsberechtigungen . HDFS unterstützt keine Hardlinks oder Softlinks. Die HDFS-Architektur schließt jedoch die Implementierung dieser Funktionen nicht aus. Während HDFS folgt Namenskonvention des Dateisystems , einige Pfade und Namen (zB /. reserviert und . Schnappschuss) sind reserviert. Funktionen wie

transparente Verschlüsselung und Schnappschuss Verwenden Sie reservierte Pfade.

Der NameNode verwaltet den Dateisystem-Namespace. Jede Änderung des Dateisystem-Namespace oder seiner Eigenschaften wird vom NameNode aufgezeichnet. Eine Anwendung kann die Anzahl der Replikate einer Datei angeben, die von HDFS verwaltet werden sollen. Die Anzahl der Kopien einer Datei wird als Replikationsfaktor dieser Datei bezeichnet. Diese Informationen werden vom NameNode gespeichert.

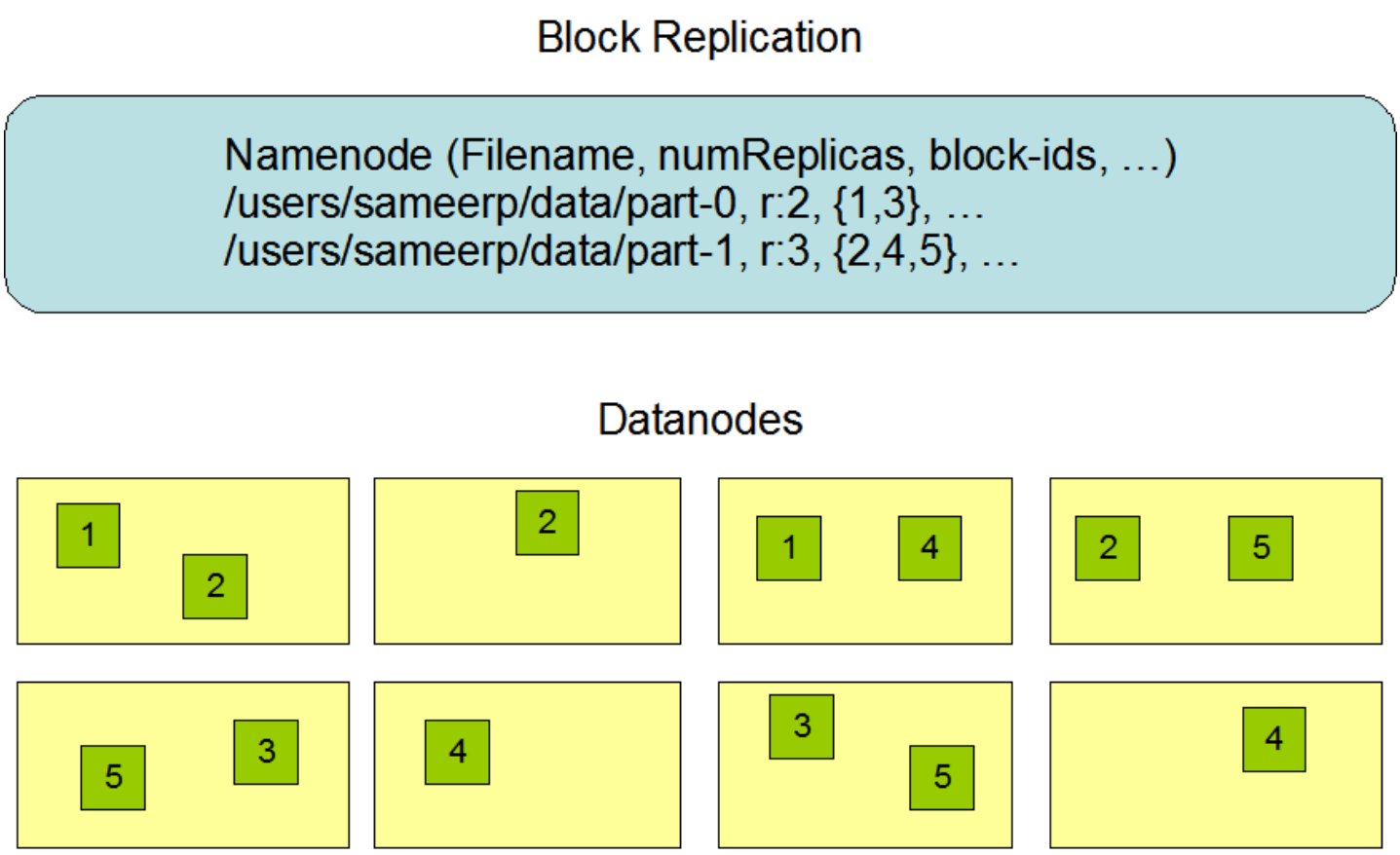
Datenreplikation

HDFS wurde entwickelt, um sehr große Dateien zuverlässig auf mehreren Computern in einem großen Cluster zu speichern. Es speichert jede Datei als eine Folge von Blöcken. Die Blöcke einer Datei werden aus Gründen der Fehlertoleranz repliziert. Die Blockgröße und der Replikationsfaktor können pro Datei konfiguriert werden.

Alle Blöcke in einer Datei mit Ausnahme des letzten Blocks haben dieselbe Größe, während Benutzer einen neuen Block starten können, ohne den letzten Block auf die konfigurierte Blockgröße auszufüllen, nachdem die Unterstützung für Blöcke variabler Länge zu append und hsync hinzugefügt wurde.

Eine Anwendung kann die Anzahl der Replikate einer Datei angeben. Der Replikationsfaktor kann zum Zeitpunkt der Dateierstellung angegeben und später geändert werden. Dateien in HDFS werden einmal geschrieben (mit Ausnahme von Anhängen und Abschneiden) und haben jeweils nur einen Writer.

Der NameNode trifft alle Entscheidungen bezüglich der Replikation von Blöcken. Es empfängt regelmäßig einen Heartbeat und einen Blockreport von jedem der DataNodes im Cluster. Der Empfang eines Heartbeat bedeutet, dass der DataNode ordnungsgemäß funktioniert. Ein Blockbericht enthält eine Liste aller Blöcke in einem DataNode.



Replica Placement: Die ersten Babyschritte

Die Platzierung von Replikaten ist für die Zuverlässigkeit und Leistung von HDFS von entscheidender Bedeutung. Durch die Optimierung der Replikatplatzierung unterscheidet sich HDFS von den meisten anderen verteilten Dateisystemen. Dies ist eine Funktion, die viel Abstimmung und Erfahrung erfordert. Der Zweck einer Rack-fähigen Replikatplatzierungsrichtlinie besteht darin, die Datenzuverlässigkeit, Verfügbarkeit und Netzwerkbandbreitennutzung zu verbessern. Die aktuelle Implementierung der Replikatplatzierungsrichtlinie ist ein erster Versuch in diese Richtung. Die kurzfristigen Ziele der Implementierung dieser Richtlinie bestehen darin, sie auf Produktionssystemen zu validieren, mehr über ihr Verhalten zu erfahren und eine Grundlage für das Testen und Erforschen komplexerer Richtlinien zu schaffen.

Große HDFS-Instanzen werden auf einem Cluster von Computern ausgeführt, die normalerweise über viele Racks verteilt sind. Die Kommunikation zwischen zwei Knoten in verschiedenen Racks muss über Switches erfolgen. In den meisten Fällen ist die **Netzwerkbandbreite zwischen Computern in demselben Rack größer als die Netzwerkbandbreite zwischen Computern in verschiedenen Racks. Der NameNode bestimmt die Rack-ID, zu der jeder DataNode gehört, über den in beschriebenen Prozess** Hadoop Rack Awareness . **Eine einfache, aber nicht optimale Richtlinie besteht darin, Replikate auf eindeutigen Racks zu platzieren. Dies verhindert den Verlust von Daten, wenn ein gesamtes Rack ausfällt, und ermöglicht die Nutzung der Bandbreite mehrerer** Racks beim Lesen von Daten. Diese Richtlinie verteilt Replikate gleichmäßig im Cluster, wodurch die Last bei Komponentenausfällen leicht ausgeglichen werden kann. Diese Richtlinie erhöht jedoch die Kosten für Schreibvorgänge, da beim Schreiben Blöcke auf mehrere Racks übertragen werden müssen.

Für den allgemeinen Fall, wenn der Replikationsfaktor drei beträgt, besteht die Platzierungsrichtlinie von HDFS darin, ein Replikat auf dem lokalen Computer abzulegen, wenn sich der Writer auf einem Datenknoten befindet, andernfalls auf einem zufälligen Datenknoten im selben Rack wie der des Writers, auf einem anderen Replikat ein Knoten in einem anderen (Remote-) Rack und der letzte auf einem anderen Knoten in demselben Remote-Rack. Diese Richtlinie reduziert den Schreibverkehr zwischen Racks, wodurch die Schreibleistung im Allgemeinen verbessert wird. Die Wahrscheinlichkeit eines Rack-Ausfalls ist weitaus geringer als die eines Knotenausfalls. Diese Richtlinie hat keine Auswirkungen auf die Zuverlässigkeit und Verfügbarkeit von Daten. Es reduziert jedoch die beim Lesen von Daten verwendete Gesamtnetzwerkbandbreite, da ein Block nur in zwei eindeutigen Racks anstatt in drei platziert wird. Mit dieser Richtlinie werden die Replikate einer Datei nicht gleichmäßig auf die Racks verteilt. Ein Drittel der Replikate befindet sich auf einem Knoten. Zwei Drittel der Replikate befinden sich auf einem Rack, und das andere Drittel ist gleichmäßig auf die verbleibenden Racks verteilt. Diese Richtlinie verbessert die Schreibleistung, ohne die Datenzuverlässigkeit oder die Leseleistung zu beeinträchtigen.

Wenn der Replikationsfaktor größer als 3 ist, wird die Platzierung der 4. und der folgenden Replikate zufällig bestimmt, während die Anzahl der Replikate pro Rack unter der Obergrenze bleibt (was im Grunde **genommen (Repliken - 1) / Gestelle + 2).**

Da der NameNode nicht zulässt, dass DataNodes mehrere Replikate desselben Blocks haben, entspricht die maximale Anzahl der erstellten Replikate der Gesamtzahl der zu diesem Zeitpunkt vorhandenen DataNodes.

Nach der Unterstützung für Speichertypen und Speicherrichtlinien Wurde HDFS hinzugefügt, berücksichtigt der NameNode die Richtlinie für die Platzierung von Replikaten zusätzlich zu der oben beschriebenen Rack-Erkennung. Der NameNode wählt zuerst Knoten basierend auf der Rack-Erkennung aus und überprüft dann, ob der Kandidatenknoten über Speicher verfügt, der von der mit der Datei verknüpften Richtlinie benötigt wird. Wenn der Kandidatenknoten nicht über den Speichertyp verfügt, sucht der NameNode nach einem anderen Knoten. Wenn im ersten Pfad nicht genügend Knoten zum Platzieren von Replikaten gefunden werden können, sucht der NameNode nach Knoten mit Fallback-Speichertypen im zweiten Pfad. Die hier beschriebene aktuelle Standardrichtlinie für die Platzierung von Replikaten ist in Arbeit.

Replikatauswahl

Um den globalen Bandbreitenverbrauch und die Leselatenz zu minimieren, versucht HDFS, eine Leseanforderung von einem Replikat zu erfüllen, das dem Lesegerät am nächsten liegt. Wenn sich auf demselben Rack wie der Leseknoten ein Replikat befindet, wird dieses Replikat bevorzugt, um die Leseanforderung zu erfüllen. Wenn sich der HDFS-Cluster über mehrere Rechenzentren erstreckt, wird ein Replikat, das sich im lokalen Rechenzentrum befindet, einem Remote-Replikat vorgezogen.

Sicherheitsmodus

Beim Start wechselt der NameNode in einen speziellen Status namens Safemode. Die Replikation von Datenblöcken erfolgt nicht, wenn sich der NameNode im Safemode-Status befindet. Der NameNode empfängt Heartbeat- und Blockreport-Nachrichten von den DataNodes. Ein Blockbericht enthält die Liste der Datenblöcke, die ein DataNode hostet. Jeder Block hat eine festgelegte Mindestanzahl von Replikaten. Ein Block gilt als sicher repliziert, wenn die Mindestanzahl von Replikaten dieses Datenblocks mit dem NameNode eingecheckt wurde. Nachdem ein konfigurierbarer Prozentsatz sicher replizierter Datenblöcke beim NameNode eingecheckt hat (plus weitere 30 Sekunden), verlässt der NameNode den Safemode-Status. Anschließend wird die Liste der Datenblöcke (falls vorhanden) ermittelt, die immer noch weniger als die angegebene Anzahl von Replikaten aufweisen. Der NameNode repliziert diese Blöcke dann auf andere DataNodes.

Die Persistenz von Dateisystem-Metadaten

Der HDFS-Namespace wird vom NameNode gespeichert. Der NameNode verwendet ein Transaktionsprotokoll namens EditLog, um jede Änderung an Dateisystemmetadaten dauerhaft aufzuzeichnen. Wenn Sie beispielsweise eine neue Datei in HDFS erstellen, fügt der NameNode einen Datensatz in das EditLog ein, der dies anzeigt. In ähnlicher Weise wird durch Ändern des Replikationsfaktors einer Datei ein neuer Datensatz in das EditLog eingefügt. Der NameNode verwendet eine Datei in seinem lokalen Host-Betriebssystem-Dateisystem, um das EditLog zu speichern. Der gesamte Dateisystem-Namespace, einschließlich der Zuordnung von Blöcken zu Dateien und Dateisystemeigenschaften, wird in einer Datei namens FsImage gespeichert. Das FsImage wird auch als Datei im lokalen Dateisystem des NameNode gespeichert.

Der NameNode speichert ein Image des gesamten Dateisystem-Namespace und der Datei Blockmap im Speicher. Wenn der NameNode gestartet wird oder ein Prüfpunkt durch einen konfigurierbaren Schwellenwert ausgelöst wird, liest er FsImage und EditLog von der Festplatte, wendet alle Transaktionen aus dem EditLog auf die speicherinterne Darstellung des FsImage an und löscht diese neue Version in a neues FsImage auf der Festplatte. Es kann dann das alte EditLog abschneiden, da seine Transaktionen auf das persistente FsImage angewendet wurden. Dieser Vorgang wird als Checkpoint bezeichnet. Mit einem Prüfpunkt soll sichergestellt werden, dass HDFS eine konsistente Ansicht der Dateisystemmetadaten hat, indem ein Snapshot der Dateisystemmetadaten erstellt und in FsImage gespeichert wird. Obwohl es effizient ist, ein FsImage zu lesen, ist es nicht effizient, inkrementelle Änderungen direkt an einem FsImage vorzunehmen. Anstatt FsImage für jede Bearbeitung zu ändern, behalten wir die Änderungen im Bearbeitungsprotokoll bei. Während des Prüfpunkts werden die Änderungen von Editlog auf das FsImage angewendet. Ein Prüfpunkt kann in einem bestimmten Zeitintervall ausgelöst werden (dfs.namenode.checkpoint.period) ausgedrückt in Sekunden oder nachdem sich eine bestimmte Anzahl von Dateisystemtransaktionen angesammelt hat (dfs.namenode.checkpoint.txns). Wenn beide Eigenschaften festgelegt sind, löst der erste zu erreichende Schwellenwert einen Prüfpunkt aus.

Der DataNode speichert HDFS-Daten in Dateien in seinem lokalen Dateisystem. Der DataNode kennt keine HDFS-Dateien. Jeder HDFS-Datenblock wird in einer separaten Datei in seinem lokalen Dateisystem gespeichert. Der DataNode erstellt nicht alle Dateien im selben Verzeichnis. Stattdessen wird eine Heuristik verwendet, um die optimale Anzahl von Dateien pro Verzeichnis zu ermitteln und Unterverzeichnisse entsprechend zu erstellen. Es ist nicht optimal, alle lokalen Dateien im selben Verzeichnis zu erstellen, da das lokale Dateisystem möglicherweise eine große Anzahl von Dateien in einem einzelnen Verzeichnis nicht effizient unterstützen kann. Wenn ein DataNode gestartet wird, durchsucht er sein lokales Dateisystem, generiert eine Liste aller HDFS-Datenblöcke, die jeder dieser lokalen Dateien entsprechen, und sendet diesen Bericht an den NameNode. Der Bericht heißt *Blockbericht*.

Die Kommunikationsprotokolle

Alle HDFS-Kommunikationsprotokolle werden über das TCP / IP-Protokoll gelegt. Ein Client stellt eine Verbindung zu einem konfigurierbaren TCP-Port auf dem NameNode-Computer her. Es spricht das ClientProtocol mit dem NameNode. Die DataNodes kommunizieren über das DataNode-Protokoll mit dem NameNode. Eine RPC-Abstraktion (Remote Procedure Call) umschließt sowohl das Client-Protokoll als auch das DataNode-Protokoll. Der NameNode initiiert standardmäßig keine RPCs. Stattdessen werden nur RPC-Anforderungen von DataNodes oder Clients beantwortet.

Robustheit

Das Hauptziel von HDFS ist es, Daten auch bei Fehlern zuverlässig zu speichern. Die drei häufigsten Arten von Fehlern sind NameNode-Fehler, DataNode-Fehler und Netzwerkpartitionen.

Datenfestplattenfehler, Heartbeats und erneute Replikation

Jeder DataNode sendet regelmäßig eine Heartbeat-Nachricht an den NameNode. Eine Netzwerkpartition kann dazu führen, dass eine Teilmenge von DataNodes die Verbindung zum NameNode verliert. Der NameNode erkennt diesen Zustand durch das Fehlen einer Heartbeat-Nachricht. Der NameNode markiert DataNodes ohne letzten Heartbeats als tot und leitet keine neuen E / A-Anforderungen an sie weiter. Alle Daten, die in einem toten DataNode registriert wurden, stehen HDFS nicht mehr zur Verfügung. Der Tod von DataNode kann dazu führen, dass der Replikationsfaktor einiger Blöcke unter den angegebenen Wert fällt. Der NameNode verfolgt ständig, welche Blöcke repliziert werden müssen, und initiiert bei Bedarf die Replikation. Die Notwendigkeit einer erneuten Replikation kann aus vielen Gründen auftreten: Ein DataNode ist möglicherweise nicht mehr verfügbar, ein Replikat ist möglicherweise beschädigt, eine Festplatte auf einem DataNode kann ausfallen.

Die Zeitüberschreitung zum Markieren von DataNodes ist konservativ lang (standardmäßig über 10 Minuten), um einen Replikationssturm zu vermeiden, der durch das Flattern von DataNodes verursacht wird. Benutzer können kürzere Intervalle festlegen, um DataNodes als veraltet zu markieren und veraltete Knoten beim Lesen und / oder Schreiben durch Konfiguration für leistungsabhängige Workloads zu vermeiden.

Cluster-Rebalancing

Die HDFS-Architektur ist mit Datenausgleichsschemata kompatibel. Ein Schema kann Daten automatisch von einem DataNode auf einen anderen verschieben, wenn der freie Speicherplatz auf einem DataNode einen bestimmten Schwellenwert unterschreitet. Im Falle einer plötzlich hohen Nachfrage nach einer bestimmten Datei kann ein Schema dynamisch zusätzliche Replikate erstellen und andere Daten im Cluster neu ausgleichen. Diese Arten von Datenausgleichsschemata sind noch nicht implementiert.

Datenintegrität

Es ist möglich, dass ein Datenblock, der von einem DataNode abgerufen wird, beschädigt ankommt. Diese Beschädigung kann aufgrund von Fehlern in einem Speichergerät, Netzwerkfehlern oder fehlerhafter Software auftreten. Die HDFS-Client-Software implementiert eine Prüfsummenprüfung für den Inhalt von HDFS-Dateien. Wenn ein Client eine HDFS-Datei erstellt, berechnet er eine Prüfsumme für jeden Block der Datei und speichert diese Prüfsummen in einer separaten versteckten Datei im selben HDFS-Namespace. Wenn ein Client den Dateiinhalt abruft, überprüft er, ob die von jedem DataNode empfangenen Daten mit der in der zugehörigen Prüfsummendatei gespeicherten Prüfsumme übereinstimmen. Wenn nicht, kann der Client diesen Block von einem anderen DataNode abrufen, der eine Replik dieses Blocks enthält.

Metadaten-Datenträgerfehler

Das FsImage und das EditLog sind zentrale Datenstrukturen von HDFS. Eine Beschädigung dieser Dateien kann dazu führen, dass die HDFS-Instanz nicht mehr funktioniert. Aus diesem Grund kann der NameNode so konfiguriert werden, dass mehrere Kopien von FsImage und EditLog verwaltet werden. Bei jeder Aktualisierung von FsImage oder EditLog werden alle FsImages und EditLogs synchron aktualisiert. Diese synchrone Aktualisierung mehrerer Kopien von FsImage und EditLog kann die Rate der Namespace-Transaktionen pro Sekunde verringern, die ein NameNode unterstützen kann. Diese Verschlechterung ist jedoch akzeptabel, da HDFS-Anwendungen zwar sehr datenintensiv sind, jedoch nicht metadatenintensiv. Beim Neustart eines NameNode werden die neuesten konsistenten FsImage- und EditLog-Dateien ausgewählt. gemeinsamer Speicher auf NFS oder mit a

verteiltes Bearbeitungsprotokoll (genannt Journal). Letzteres ist der empfohlene Ansatz.

Schnappschüsse

Schnappschüsse Unterstützung beim Speichern einer Kopie von Daten zu einem bestimmten Zeitpunkt. Eine Verwendung der Snapshot-Funktion kann darin bestehen, eine beschädigte HDFS-Instanz auf einen zuvor bekannten guten Zeitpunkt zurückzusetzen.

Datenorganisation

Datenblöcke

HDFS unterstützt sehr große Dateien. Mit HDFS kompatible Anwendungen sind solche, die sich mit großen Datenmengen befassen. Diese Anwendungen schreiben ihre Daten nur einmal, lesen sie jedoch ein- oder mehrmals und erfordern, dass diese Lesevorgänge bei Streaming-Geschwindigkeiten erfüllt werden. HDFS unterstützt die Semantik zum Schreiben und einmaligen Lesen von Dateien. Eine typische von HDFS verwendete Blockgröße beträgt 128 MB. Somit wird eine HDFS-Datei in 128-MB-Blöcke zerlegt, und wenn möglich befindet sich jeder Block auf einem anderen DataNode.

Replikations-Pipelining

Wenn ein Client Daten in eine HDFS-Datei mit einem Replikationsfaktor von drei schreibt, ruft der NameNode eine Liste von DataNodes mithilfe eines Algorithmus zur Auswahl des Replikationsziels ab. Diese Liste enthält die DataNodes, die ein Replikat dieses Blocks hosten. Der Client schreibt dann in den ersten DataNode. Der erste DataNode beginnt mit dem Empfang der Daten in Teilen, schreibt jeden Teil in sein lokales Repository und überträgt diesen Teil an den zweiten DataNode in der Liste. Der zweite DataNode beginnt wiederum mit dem Empfang jedes Teils des Datenblocks, schreibt diesen Teil in sein Repository und löscht diesen Teil dann in den dritten DataNode. Schließlich schreibt der dritte DataNode die Daten in sein lokales Repository. Somit kann ein DataNode Daten von dem vorherigen in der Pipeline empfangen und gleichzeitig Daten an den nächsten in der Pipeline weiterleiten. Somit,

Barrierefreiheit

Auf HDFS kann von Anwendungen auf viele verschiedene Arten zugegriffen werden. HDFS bietet von Haus aus a Dateisystem Java API für Anwendungen zu verwenden. EIN C-Wrapper für diese Java API und REST-API ist ebenfalls verfügbar. Darüber hinaus ein HTTP-Browser und kann auch zum Durchsuchen der Dateien einer HDFS-Instanz verwendet werden. Durch die Nutzung NFS-Gateway , HDFS kann als Teil des lokalen Dateisystems des Clients bereitgestellt werden.

FS Shell

Mit HDFS können Benutzerdaten in Form von Dateien und Verzeichnissen organisiert werden. Es bietet eine Befehlszeilenschnittstelle namens FS-Shell Dadurch kann ein Benutzer mit den Daten in HDFS interagieren. Die Syntax dieses Befehlssatzes ähnelt anderen Shells (z. B. bash, csh), mit denen Benutzer bereits vertraut sind. Hier sind einige Beispiele für Aktions- / Befehlspaare:

Aktion	Befehl
Erstellen Sie ein Verzeichnis mit dem Namen / foodir	bin / hadoop dfs -mkdir / foodir
Entfernen Sie ein Verzeichnis mit dem Namen / foodir	bin / hadoop fs -rm -R / foodir
Zeigen Sie den Inhalt einer Datei mit dem Namen / an foodir / myfile.txt	bin / hadoop dfs -cat /foodir/myfile.txt

Die FS-Shell richtet sich an Anwendungen, die eine Skriptsprache benötigen, um mit den gespeicherten Daten zu interagieren.

DFSAdmin

Der DFSAdmin-Befehlssatz wird zum Verwalten eines HDFS-Clusters verwendet. Dies sind Befehle, die nur von einem HDFS-Administrator verwendet werden. Hier sind einige Beispiele für Aktions- / Befehlspaare:

Aktion	Befehl
Stellen Sie den Cluster in den Safemode	bin / hdfs dfsadmin -safemode eingeben
Generieren Sie eine Liste von DataNodes	bin / hdfs dfsadmin -report
Wiederinbetriebnahme oder Stilllegung von DataNode (s)	bin / hdfs dfsadmin -refreshNodes

Browser-Oberfläche

Bei einer typischen HDFS-Installation wird ein Webserver so konfiguriert, dass der HDFS-Namespace über einen konfigurierbaren TCP-Port verfügbar gemacht wird. Auf diese Weise kann ein Benutzer im HDFS-Namespace navigieren und den Inhalt seiner Dateien mithilfe eines Webbrowsers anzeigen.

Raumgewinnung

Datei löscht und löscht

Wenn die Papierkorbkonfiguration aktiviert ist, werden Dateien von entfernt FS Shell wird nicht sofort aus HDFS entfernt. Stattdessen verschiebt HDFS es in ein Papierkorbverzeichnis (jeder Benutzer hat sein eigenes Papierkorbverzeichnis unter / Benutzer / <Benutzername> /.Trash). Die Datei kann schnell wiederhergestellt werden, solange sie im Papierkorb bleibt. Die zuletzt gelöschten Dateien werden in das aktuelle Papierkorbverzeichnis verschoben (/ Benutzer / <Benutzername> /.Trash/Current), In einem konfigurierbaren Intervall erstellt HDFS Prüfpunkte (unter / Benutzer / <Benutzername> /.Trash/ <Datum>) für Dateien im aktuellen Papierkorbverzeichnis und löscht alte Prüfpunkte, wenn sie abgelaufen sind. Sehen Befehl zum Löschen der FS-Shell über Checkpointing von Müll.

Nach Ablauf seiner Lebensdauer im Papierkorb löscht der NameNode die Datei aus dem HDFS-Namespace. Durch das Löschen einer Datei werden die mit der Datei verknüpften Blöcke freigegeben. Beachten Sie, dass zwischen dem Löschen einer Datei durch einen Benutzer und dem Zeitpunkt der entsprechenden Erhöhung des freien Speicherplatzes in HDFS eine erhebliche Zeitverzögerung auftreten kann.

Das folgende Beispiel zeigt, wie die Dateien von FS Shell aus HDFS gelöscht werden. Wir haben 2 Dateien (test1 & test2) unter dem Verzeichnis delete erstellt

```
$ hadoop fs -mkdir -p delete / test1 $ hadoop fs -mkdir -p delete /
test2 $ hadoop fs -ls delete / Found 2 items


drwxr-xr-x - hadoop hadoop                0 2015-05-08 12:39 delete / test1
drwxr-xr-x - hadoop hadoop                0 2015-05-08 12:40 delete / test2
```

Wir werden die Datei test1 entfernen. Der folgende Kommentar zeigt, dass die Datei in das Papierkorbverzeichnis verschoben wurde.

```
$ hadoop fs -rm -r delete / test1
Verschoben: hdfs: // localhost: 8020 / user / hadoop / delete / test1 in den Papierkorb unter: hdfs: // localhost: 8020 / user / hadoop / .Trash
```

Jetzt werden wir die Datei mit der Option skipTrash entfernen, wodurch die Datei nicht an den Papierkorb gesendet wird. Sie wird vollständig aus HDFS entfernt.

```
$ hadoop fs -rm -r -skipTrash delete / test2 Gelöscht delete / test2
```

Wir können jetzt sehen, dass das Papierkorbverzeichnis nur die Datei test1 enthält.

```
$ hadoop fs -ls .Trash / Current / user / hadoop / delete / 1 Elemente gefunden \

drwxr-xr-x - hadoop hadoop                0 2015-05-08 12:39 .Trash / Current / user / hadoop / delete / test1
```

Die Datei test1 wird also in den Papierkorb verschoben und die Datei test2 wird dauerhaft gelöscht.

Replikationsfaktor verringern

Wenn der Replikationsfaktor einer Datei reduziert wird, wählt der NameNode überschüssige Replikate aus, die gelöscht werden können. Der nächste Heartbeat überträgt diese Informationen an den DataNode. Der DataNode entfernt dann die entsprechenden Blöcke und der entsprechende freie Speicherplatz wird im Cluster angezeigt. Auch hier kann es zu einer Zeitverzögerung zwischen dem Abschluss des Aufrufs der setReplication-API und dem Auftreten von freiem Speicherplatz im Cluster kommen.

Verweise

Hadoop JavaDoc-API .

HDFS-Quellcode: http://hadoop.apache.org/version_control.html