Das Hadoop Okosystem

Das Hadoop Ökosystem

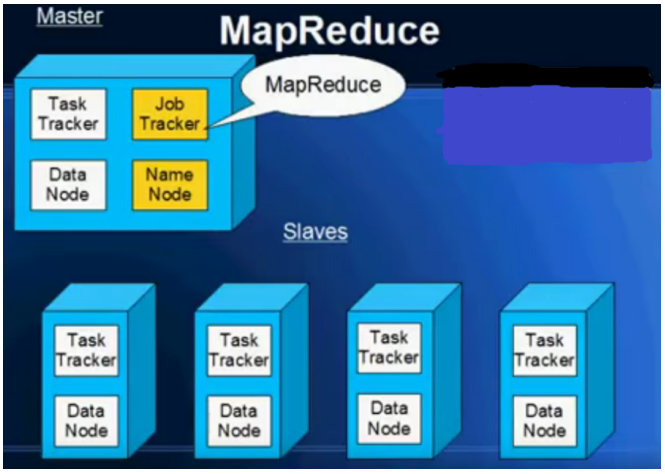
Allgemein

Die wichtigsten Komponenten von Hadoop und ihre Einordnung im Ökosystem sind auf der Abbildung dargestellt

Das **Hadoop Distributed File System** (**HDFS**) wird oft als Grundlage für den Rest des Hadoop-Ökosystems angesehen. HDFS ist die **Speicher schicht** für **Hadoop** und bietet die Möglichkeit, große Datenmengen zu speichern und gleichzeitig die Speicherkapazität zu erhöhen und die Bandbreite linear zu aggregieren. **HDFS** ist ein **logisches** Dateisystem, das viele Server mit jeweils mehreren Festplatten umfasst. Dies ist aus Sicherheitsgründen wichtig, da eine bestimmte Datei in HDFS mehrere oder alle Server im Hadoop-Cluster umfassen kann. Dies bedeutet, dass Client-Interaktionen mit einer bestimmten Datei möglicherweise die Kommunikation mit jedem Knoten im Cluster erfordern. Dies wird durch eine Schlüsselimplementierungsfunktion von **HDFS ermöglicht, die Dateien in Blöcke aufteilt**. Jeder Datenblock für eine bestimmte Datei kann auf einem beliebigen physischen Laufwerk, auf einem beliebigen Knoten im Cluster gespeichert werden. Der wichtige Sicherheitsaspekt besteht darin, dass **alle Dateien in HDFS in Blöcke aufgeteilt** sind und Clients, die HDFS verwenden, beim Lesen und Schreiben von Dateien über das Netzwerk mit allen Servern im Hadoop-Cluster kommunizieren.

- **Apache Hadoop**
- Die Komponenten von Hadoop sind bei der Apache Foundation angesiedelt und stehen unter der Apache License 2.0. Bei Hadoop handelt es sich nicht nur um ein einzelnes Projekt, sondern um eine Vielzahl von zusammenhängenden und aufeinander aufbauenden Projekten. In diesem Zusammenhang spricht man gerne von den Kernkomponenten von Hadoop, welche die Grundlage für Hadoop und alle anderen Projekte bilden, die man als das Hadoop Ökosystem bezeichnet. Die Kernprojekte sind die folgenden:
- **Hadoop Distributed File System (HDFS):** Ein verteiltes Dateisystem für die Speicherung und Verwaltung großer Datenmengen
- **Hadoop MapReduce:** Ein Framework zur verteilten und parallelen Verarbeitung von in HDFS gespeicherten Datenmengen
- **Apache Mahout**
- Mahout ist eine skalierbare, für Hadoop entwickelte und sehr mächtige Bibliothek für Machine Learning und Data Mining. Das Projekt beinhaltet die Implementierung diverser Machine Learning Algorithmen, unter anderem zur Klassifizierung, dem Clustering und dem Collaborative Filtering. Auf Basis von Mahout lassen sich so z.B. mächtige Recommendation Systems entwickeln, sie wissen schon, diese Subsysteme „Kunden, die das kauften, kaufen auch dies" in Onlineshops mit denen etwa Amazon heutzutage einen beträchtlichen Teil seines Umsatzes macht.
- **Apache MapReduce**

- **MapReduce** ist das Verarbeitungsgegenstück zu HDFS und bietet den grundlegendsten Mechanismus für die Stapelverarbeitung von Daten. Wenn MapReduce **über YARN ausgeführt wird**, wird es oft **MapReduce2** oder **MR2 genannt**. Dies unterscheidet die YARN-basierte Version von MapReduce aus dem **eigenständigen MapReduce-Framework**, das rückwirkend **MR1 genannt wurde**. MapReduce-Jobs werden von Clients an das MapReduce-Framework übergeben und arbeiten über eine Teilmenge von Daten in HDFS, normalerweise ein bestimmtes Verzeichnis. MapReduce selbst ist ein Programmierparadigma, dass es ermöglicht, Datenblöcke oder Blöcke im Fall von HDFS parallel und unabhängig voneinander von mehreren Servern zu verarbeiten. Während ein Hadoop-Entwickler die Feinheiten der Funktionsweise von MapReduce kennen muss, tut dies ein Sicherheitsarchitekt größtenteils nicht. Was ein Sicherheitsarchitekt wissen muss, ist, dass Clients ihre Aufträge an das MapReduce-Framework senden und von diesem Zeitpunkt an das MapReduce-Framework die Verteilung und Ausführung des Client-Codes im gesamten Cluster übernimmt. Clients interagieren nicht mit einem der Knoten im Cluster, um deren Job auszuführen. Jobs selbst erfordern einige Aufgaben, die ausgeführt werden müssen, um die Arbeit abzuschließen. Jeder Task wird auf einem bestimmten Knoten vom Zeitplanungsalgorithmus des MapReduce-Frameworks gestartet.
- Ein wichtiger Punkt bei MapReduce ist, dass andere Hadoop-Ökosystemkomponenten Frameworks und Bibliotheken über MapReduce sind, was bedeutet, dass **MapReduce die eigentliche Verarbeitung von Daten übernimmt**, diese **Frameworks und Bibliotheken jedoch die MapReduce-Jobausführung von Clients abstrahieren**. **Hive**, **Pig** und **Sqoop** sind Beispiele für Komponenten, die MapReduce auf diese Weise verwenden.
- **Apache Pig**: MapReduce-Jobs in Java zu schreiben ist auf Dauer aufwändig und erfordert Programmierkenntnisse. Wäre es nicht schön, wenn es eine Abstraktionsschicht dazu gäbe? Genau das dachten sich auch die Leute bei Yahoo und das Ergebnis nennt sich Apache Pig. Dabei besteht Pig aus einer abstrakten Skriptsprache (*Pig Latin*), welche es erlaubt auf einer abstrakteren Ebene eher den Datenfluss eines MapReduce-Jobs zu beschreiben. Darauf basierend erzeugt der Pig-Compiler bereits optimierte MapReduce-Jobs, welche dann wieder in gewohnter Form wieder mittels MapReduce ausgeführt werden – aber eben ohne dass der Benutzer alle Details selbst implementieren muss. Damit gibt man zwar eine komplett feingranulare Kontrolle über die Jobs auf, aber in der Praxis braucht man diese sehr selten und entsprechend wird Pig sehr häufig eingesetzt.
- **Apache Hive**:Apache Hive ist ebenfalls eine Abstraktionsschicht, welche auf dem MapReduce-Framework basiert und gerne als das als Data Warehouse System für Hadoop beschrieben wird. Hive bringt eine SQL-ähnliche Sprache (HiveQL) mit, welche es ermöglicht Aufgaben wie Aggregationen, Analysen und weitere Abfragen auf in HDFS (oder anderen Hadoop- kompatiblen File Systemen) gespeicherter Datenmengen durchzuführen. Am Ende wird aber auch HiveQL wieder in MapReduce-Jobs transformiert und so ausgeführt. Der große Vorteil ist aber auch hier die Abstraktion und die Tatsache, dass SQL-Kenntnisse weit verbreitet sind und somit auch keine reinen Techniker z.B. in Fachabteilungen von großen Firmen damit gut arbeiten können – ein wichtiger Faktor für die große Verbreitung im Hadoop-Umfeld.



- **Apache HCatalog**
- HCatalog verbindet Welten – in der Praxis vor allem Pig und Hive. Es stellt einen zentralen Tabellen- und Metadatenmanagementservice zur Beschreibung der Struktur von in Hadoop abgelegten Daten zur Verfügung. Einmal beschrieben können die Daten sowohl in Pig als auch in Hive bequemer verwendet werden und es lassen sich viel leichter ganze Ketten an MapReduce-Jobs aufbauen. So kann man etwa Pig für ETL-Prozesse nutzen, die Daten also importieren und bereinigen und sie dann mit Hive weiterverarbeiten – ein in der Praxis häufig angetroffener Prozess.
- **Apache HBase**
- HBase ist eine Open-Source-, nicht-relationale, verteilte **Datenbank, die** nach Googles BigTable und in **Java geschrieben ist**. Es läuft **auf HDFS** (Hadoop Distributed Filesystem) und bietet BigTable ähnliche Funktionen für Hadoop. **HBase** bietet Komprimierung, In-Memory-Operation und Bloom-Filter pro Spalte. Tabellen in HBase können als Eingabe und Ausgabe für MapReduce-Jobs dienen, die in Hadoop ausgeführt werden. Auf sie kann über die Java-API, aber auch über REST-, Avro- oder Thrift-Gateway-APIs zugegriffen werden. HBase ist ein spaltenorientierter Schlüssel - Wertdatenspeicher. HBase verwendet normalerweise HDFS als zugrundeliegende Speicherschicht für Daten.
- **Apache Sqoop**
- Apache Sqoop bietet die Möglichkeit, Daten von und zu einem traditionellen **R**elational**D**ata**B**ase**M**anagement**S**ystem sowie anderen Datenquellen wie FTP-Servern zu importieren und zu exportieren. Sqoop übergibt selbst MapReduce-Jobs, die Aufgaben zur parallelen Interaktion mit dem RDBMS starten. Sqoop wird sowohl als einfacher Mechanismus zum anfänglichen Start eines Hadoop-Clusters mit Daten als auch als Werkzeug für regelmäßige Routinen zur Aufnahme und Extraktion verwendet. Sqoop1 ist eine Gruppe von Clientbibliotheken, die über die Befehlszeile mithilfe der sqoop-Binärdatei aufgerufen werden.
- **Apache Flume**
- Apache Flume ist ein ereignisbasiertes Ingestion-Tool, das hauptsächlich für die Aufnahme in Hadoop verwendet wird, aber tatsächlich vollständig unabhängig davon verwendet werden kann. Flume, wie der Name schon vermuten lässt, wurde ursprünglich für die Aufnahme von Log-Ereignissen in HDFS erstellt. Die Flume-Architektur besteht aus drei Hauptteilen: Quellen, Senken und Kanälen.

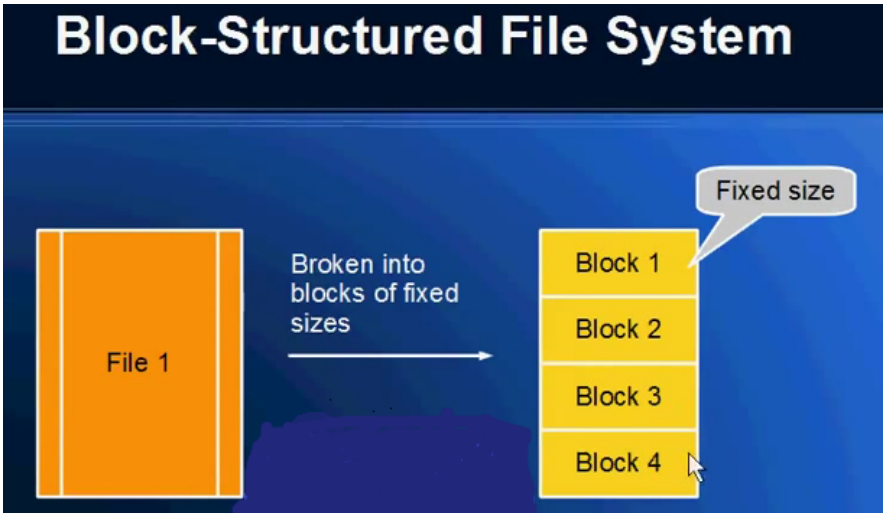
Eine Flume-Quelle definiert, wie Daten vom Upstream-Provider gelesen werden sollen. Dazu gehören Dinge wie ein Syslog-Server, eine JMS-Warteschlange oder sogar ein Polling eines Linux-Verzeichnisses. Eine Flume-Senke definiert, wie Daten downstream geschrieben werden sollen. Normal gehört zu einer Flume Senke eine HDFS-Senke und eine HBase-Senke. Schließlich definiert ein Flume-Kanal, wie Daten zwischen der Quelle und der Senke gespeichert werden. Die zwei primären Flume-Kanäle sind der Speicherkanal und der Dateikanal. Der Speicherkanal bietet Geschwindigkeit auf Kosten der Zuverlässigkeit, und der Dateikanal bietet Zuverlässigkeit auf Kosten der Geschwindigkeit. Flume besteht aus einer einzigen Komponente, einem Flume-Mittel. Agenten enthalten den Code für Quellen, Senken und Kanäle. Ein wichtiger Teil der Flume-Architektur besteht darin, dass Flume-Agenten miteinander verbunden werden können, wobei die Senke eines Agenten mit der Quelle eines anderen Agenten verbunden ist.

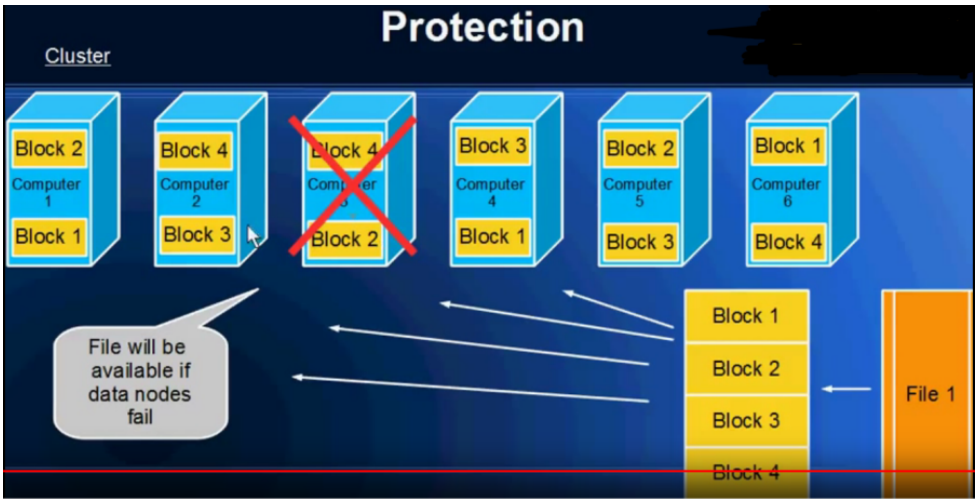
- **Apache Oozie**
- Apache Oozie ist ein **Workflow-Management-** und **Orchestrierungssystem** für Hadoop. Es ermöglicht die Einrichtung von Workflows mit verschiedenen Aktionen, von denen jede eine andere Komponente im Hadoop-Ökosystem nutzen kann. Zum Beispiel könnte ein Oozie-Workflow mit einem Sqoop-Import beginnen, um Daten in HDFS zu verschieben, dann mit einem Pig-Skript, um die Daten zu transformieren, gefolgt von einem Hive-Skript, um Metadatenstrukturen einzurichten. Oozie ermöglicht komplexere Workflows, z. B. Forks und Joins, bei denen mehrere Schritte parallel ausgeführt werden können, und andere Schritte, die auf mehreren Schritten basieren, bevor sie fortgesetzt werden. Oozie-Workflows können mit einem wiederholbaren Zeitplan ausgeführt werden, der auf verschiedenen Arten von Eingabebedingungen basiert, z. B. zu einer bestimmten Zeit ausgeführt wird oder bis ein bestimmter Pfad in HDFS existiert. Oozie besteht aus nur einer einzigen Serverkomponente und dieser Server ist für die Verarbeitung von Client-Workflows verantwortlich, verwaltet die Ausführung von Workflows und meldet den Status.
- **Apache Ambari**
- Eine der Herausforderungen bei Hadoop ist sicherlich die Installation, Administration und das Monitoring des Clusters, da dieser durchaus mal aus tausenden von Knoten bestehen kann. War dies lange Zeit eine Domäne der Cloudera Distribution mit ihrem kommerziellen Cloudera Manager, gibt es mittlerweile mit Apache Ambari ein quelloffenes und ausgereiftes System, welches genau diese Aufgaben mittels einer intuitiven Weboberfläche löst.
- **Apache ZooKeeper**
- Apache ZooKeeper ist ein **verteilter Koordinationsdienst**, der es verteilten Systemen ermöglicht, kleine Datenmengen synchron zu speichern und zu lesen. Es wird oft zum Speichern allgemeiner Konfigurationsinformationen verwendet. Darüber hinaus wird ZooKeeper im Hadoop-Ökosystem zur Synchronisierung von Hochverfügbarkeitsdiensten wie NameNode HA und ResourceManager HA stark eingesetzt. ZooKeeper selbst ist ein verteiltes System, das auf einer ungeraden Anzahl von **Servern basiert**, die als **ZooKeeper-Ensemble bezeichnet werden**, um ein Quorum oder eine Mehrheit zu erreichen, um eine gegebene Transaktion zu bestätigen. ZooKeeper hat nur eine Komponente, den ZooKeeper Server.

- **Apache Schwarm**
- Das Apache Hive-Projekt wurde von Facebook gestartet. Das Unternehmen erkannte den Nutzen von MapReduce für die Verarbeitung von Daten, fand jedoch aufgrund der fehlenden Java-Programmierkenntnisse in seinen Analystengemeinschaften Einschränkungen bei der Einführung des Frameworks. Die meisten Facebook-Analysten verfügten über SQL-Kenntnisse. Daher wurde das Hive-Projekt als **S QL-Abstraktionsschicht** gestartet, die MapReduce als Ausführungsmodul verwendet. **Apache Sentry** Sentry ist die Komponente, die einigen anderen Ökosystemkomponenten wie **Hive** und **Impala** feingranulare rollenbasierte Zugriffskontrollen (**RBAC**) bietet. Während einzelne Komponenten ihren eigenen Autorisierungsmechanismus haben können, bietet Sentry eine einheitliche Autorisierung, die eine zentralisierte Richtliniendurchsetzung über mehrere Komponenten ermöglicht. Es ist eine wichtige Komponente der Hadoop-Sicherheit.
- **Überwachungsserver** Der Sentry-Server ist ein Daemon-Prozess, der Policy-Lookups anderer Hadoop-Ökosystemkomponenten ermöglicht. Client-Komponenten von Sentry sind so konfiguriert, dass Autorisierungsentscheidungen basierend auf den Richtlinien von Sentry delegiert werden.
- **Richtliniendatenbank** Die Sentry-Policy-Datenbank ist der Ort, an dem alle Autorisierungsrichtlinien gespeichert sind. Der Sentry-Server ermittelt anhand der Richtliniendatenbank, ob ein Benutzer eine bestimmte Aktion ausführen darf. Insbesondere sucht der Sentry-Server nach einer übereinstimmenden Richtlinie, die den Zugriff auf eine Ressource für den Benutzer gewährt. In früheren Versionen von Sentry war die Richtliniendatenbank eine Textdatei, die alle Richtlinien enthielt.
- **Apache Accumulo**
- Apache Accumulo ist ein sortierter und verteilter Schlüssel / Wert-Speicher, der als robustes, skalierbares, leistungsstarkes Speicher- und Retrieval-System konzipiert wurde. Wie HBase basierte Accumulo ursprünglich auf dem Google BigTable-Design, wurde aber auf dem Apache Hadoop-Ökosystem von Projekten (insbesondere HDFS, ZooKeeper und Apache Thrift) aufgebaut. **Accumulo verwendet ungefähr das gleiche Datenmodell wie HBase** .
- **Apache Solr**
- Das Apache Solr-Projekt und speziell SolrCloud ermöglicht das **Suchen und Abrufen** von Dokumenten, die Teil einer größeren Sammlung sind, die über mehrere physische Server hinweg erstellt wurde. Die Suche ist einer der kanonischen Anwendungsfälle für Big Data und ist eines der am häufigsten verwendeten Dienstprogramme, die von jedem verwendet werden, der auf das Internet zugreift. Solr basiert auf dem Apache Lucene-Projekt, das den Großteil der Indizierungs- und Suchfunktionen übernimmt. Solr erweitert diese Fähigkeiten um Funktionen für die Unternehmenssuche wie Facettennavigation, Zwischenspeicherung, Hervorhebung von Treffern und eine Administrationsoberfläche. Solr hat eine einzige Komponente, den Server. Es kann viele Solr-Server in einer einzigen Implementierung geben, die linear durch die von SolrCloud bereitgestellte Shard-Skalierung skaliert werden. SolrCloud bietet außerdem Replikationsfunktionen für Fehler in einer verteilten Umgebung.
- **Apache YARN**
- Ursprünglich von Apache als neugestalteter **Ressourcenmanager beschrieben**, wird YARN jetzt als großräumiges, verteiltes **Betriebss ystem** für Big-Data-Anwendungen charakterisiert.
- Andere Verarbeitungs-Frameworks und -Anwendungen wie **Impala** und **Spark** verwenden YARN als Ressourcenmanagement-Framework. Während YARN einen allgemeineren Ressourcenverwaltungsrahmen bietet, ist MapReduce immer noch die kanonische Anwendung, die darauf ausgeführt wird. MapReduce, dass auf YARN ausgeführt wird, wird als Version 2 oder kurz MR2 betrachtet.
- **Cloudera Impala**
- Cloudera Impala ist ein massives Parallelverarbeitungs-Framework (**MPP**), das speziell für analytisches SQL entwickelt wurde. Impala liest Daten aus HDFS und nutzt die Hive-Metasorte zur Interpretation von Datenstrukturen und -formaten.
- Neue Benutzer im Hadoop-Ökosystem stellen oft die Frage, was den Unterschied zwischen Hive und Impala ausmacht, da beide SQL-Zugriff auf Daten in HDFS bieten. **Hive** wurde erstellt, damit Benutzer, die mit SQL vertraut sind**, Daten in HDFS verarbeiten können,**

ohne etwas über MapReduce wissen zu müssen. Es wurde **entworfen, um die Innereien von MapReduce zu abstrahieren**, um die Daten in HDFS zugänglicher zu machen. **Hive** wird hauptsächlich für **Stapelzugriff** und **ETL-Arbeit verwendet**. Im Gegensatz dazu wurde **Impala** von Grund auf als schnelle **Analyse-Engine** zur Unterstützung von Ad-hoc-Abfragen und Business Intelligence (**BI**) - Tools konzipiert. Sowohl in Hive als auch in Impala gibt es einen Nutzen, und sie sollten als komplementäre Komponenten behandelt werden.

- **Cloudera Farbton**
- Cloudera Hue ist eine Webanwendung, die viele der Hadoop-Ökosystemkomponenten benutzerfreundlich darstellt. Hue ermöglicht einen einfachen Zugriff auf dem Hadoop-Cluster, ohne dass die Benutzer mit Linux oder den verschiedenen Befehlszeilenschnittstellen der Komponenten vertraut sein müssen. **Farbton** hat eine Anzahl unterschiedlicher Sicherheitskontrollen verfügbar. **Farbton** besteht aus den folgenden Komponenten:
- **Farbton-Server**
    - Dies ist die Hauptkomponente von Hue. Es ist effektiv ein Webserver, der den Benutzern Webinhalte bereitstellt. Benutzer werden bei der ersten Anmeldung authentifiziert und von dort aus werden die vom Endbenutzer ausgeführten Aktionen tatsächlich von Hue selbst im Auftrag des Benutzers ausgeführt. Dieses Konzept wird als Identitätswechsel bezeichnet.
- **Kerberos-Ticket-Erneuerer**
- Wie der Name andeutet, ist diese Komponente verantwortlich für die regelmäßige Erneuerung des Kerberos-Ticket-Granting-Tickets (TGT), mit dem Hue mit dem Hadoop-Cluster interagiert, wenn Kerberos im Cluster aktiviert ist.





Speicherung von Hadoop Daten – wenn ein Computer ausfällt, werden die dort vorhandenen Daten auch auf weiteren Computern vorgehalten – Block4 ist auch auf Computer 2 + 6 und Block2 ist auch auf Computer 1 + 5

- **NameNode** Der **NameNode** ist verantwortlich für **die Verfolgung aller Metadaten** in Bezug auf die Dateien in HDFS, z. B. Dateinamen, Blockpositionen, Dateiberechtigungen und Replikation. Aus Sicherheitsgründen ist es wichtig zu wissen, dass Clients von HDFS, z. B. solche, die Dateien lesen oder schreiben, **immer mit dem NameNode kommunizieren**.
- **Datenknoten** Der DataNode ist verantwortlich für das tatsächliche Speichern und Abrufen von Datenblöcken in HDFS. Clients von HDFS, die eine bestimmte Datei lesen, erfahren vom NameNode, welcher DataNode im Cluster den angeforderten Datenblock hat. Beim Schreiben von Daten ins HDFS schreiben Clients einen Datenblock in einen DataNode, der durch den NameNode festgelegt wird. Von dort richtet DataNode eine Schreib-Pipeline zu anderen Datenknoten ein, um den Schreibvorgang basierend auf dem gewünschten Replikationsfaktor abzuschließen.
- **Secondary NameNode**Der **Secondary NameNode** dient nicht wie der Name suggeriert zum Backup des *NameNode*, sondern er ist lediglich ein Helferelement, welches in periodischen Abständen die *Block Map* und das *Journal Log* des *NameNodes* zusammenführt und dann wieder auf diesen überspielt. Dadurch wird der *NameNode* entlastet und der Cluster startet nach einem Ausfall schneller.

- **Journalknoten** Der JournalNode ist eine spezielle Komponente für HDFS. Wenn HDFS für hohe Verfügbarkeit (HA - HighAvailability) konfiguriert ist, übernehmen JournalNodes die Verantwortung für das Schreiben von HDFS-Metadateninformationen. Cluster haben typischerweise eine ungerade Anzahl von JournalNodes (normalerweise drei oder fünf), um die Mehrheit sicherzustellen. Wenn beispielsweise eine neue Datei in HDFS geschrieben wird, werden die Metadaten zur Datei in jeden JournalNode geschrieben. Wenn die Mehrzahl der JournalNodes diese Information erfolgreich schreibt, wird die Änderung als dauerhaft angesehen.
- **HttpFS** HttpFS ist eine Komponente von HDFS, die Clients für die Name-Node und DataNodes einen **Proxy** bereitstellt. Dieser Proxy **ist eine REST-API** und ermöglicht Clients, mit dem Proxy zu kommunizieren, um HDFS zu verwenden, ohne direkte Verbindung zu einer der anderen Komponenten in HDFS zu haben. HttpFS wird eine Schlüsselkomponente in bestimmten Clusterarchitekturen sein.
- **NFS-Gateway** Der NFS-Gateway ermöglicht Clients, HDFS wie ein **NFS-eingehängtes Dateisystem zu verwenden**. Das NFS-Gateway ist ein tatsächlicher **Daemon-Prozess**, der die NFS-Protokollkommunikation zwischen Clients und dem zugrundeliegenden HDFS-Cluster ermöglicht. Ähnlich wie HttpFS befindet sich das NFS-Gateway zwischen HDFS und Clients und bietet daher eine Sicherheitsgrenze, die in bestimmten Clusterarchitekturen nützlich sein kann.
- **KMS** Der Hadoop **Key Management Server** (KMS) spielt eine wichtige Rolle bei der transparenten HDFS-Verschlüsselung im Ruhezustand. Sein Zweck besteht darin**, als Vermittler** zwischen HDFS-Clients, dem NameNode, und einem Schlüsselserver zu fungieren, der **Verschlüsselungsvorgänge** wie das Entschlüsseln von Datenverschlüsselungsschlüsseln und das Verwalten von Verschlüsselungszonenschlüsseln abwickelt.