Hadoop File System

Dr. Houda EL BOUHISSI

M1 GL

Préfixes multiplicatifs

- Google en 2015 : 10 Eo (10 milliards de Go)
- Facebook en 2018 : 1 Eo de données
- Facebook : 7 Po de nouvelles données par jour
- Amazon : 1 Eo de nouvelles données par jour

signe	préfixe	facteur
k	kilo	10^3
M	méga	10^6
G	giga	10^9
T	téra	10^{12}
P	péta	10^{15}
E	exa	10^{18}
Z	zetta	10^{21}

La raison est que tout est enregistré sans conception, dans l'idée que ça pourra être exploité. Certains encouragent pour que les données collectées soient pertinentes (smart data) plutôt que volumineuses.

Distribution : données et traitements

Le traitement d'aussi grandes quantités de données impose des méthodes particulières. Un SGBD classique, même haut de gamme, est dans l'incapacité de traiter autant d'informations !!!

- Répartir les données sur plusieurs machines (jusqu'à plusieurs millions d'ordinateurs) dans des *Data Centers*
  - Système de fichiers spécial permettant de ne voir qu'un seul espace pouvant contenir des fichiers gigantesques et/ou très nombreux (HDFS).
  - Bases de données spécifiques (HBase, Cassandra, ElasticSearch).

- Traitements du type « map-reduce » :
  - Algorithmes faciles à écrire.
  - Exécutions faciles à paralléliser.

Cluster

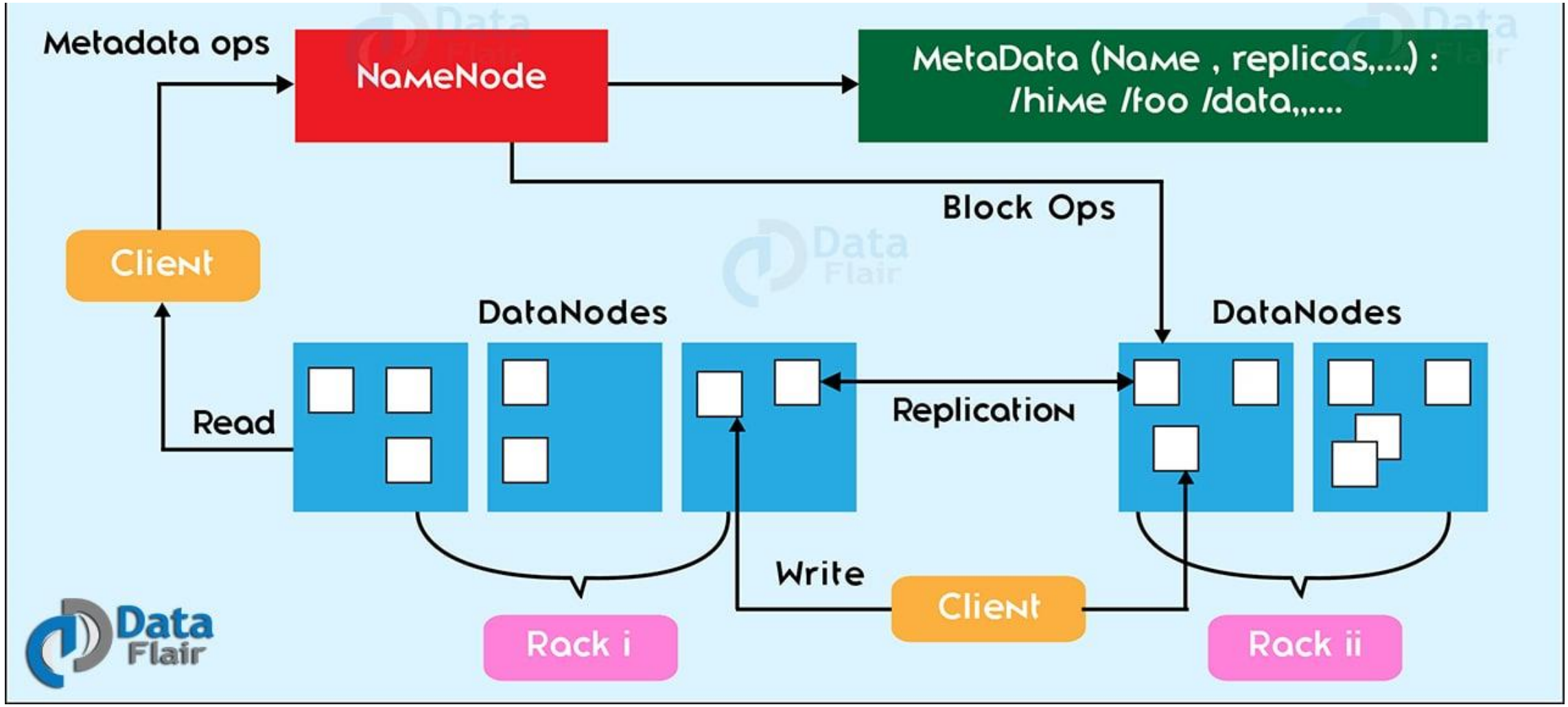
Un cluster de calcul (grappe de serveurs, imaginez 5000 ordinateurs connectés entre eux formant un cluster) est conçu spécifiquement pour stocker et analyser de grandes quantités de données non structurées au sein d'un environnement de calcul distribué.

Le travail d'analyse de données est réparti entre les différents nœuds du cluster (serveurs). Ces clusters permettent de lancer le logiciel de traitement distribué Hadoop sur des ordinateurs low-cost.

HDFS : Hadoop File System

- Système de fichiers distribué associé à Hadoop. C'est là qu'on stocke les données d'entrée, de sortie, etc.
- Les fichiers et dossiers sont organisés en arbre (comme Unix).
- Les fichiers sont stockés sur un grand nombre de machines de manière à rendre invisible la position exacte d’un fichier. L’accès est transparent, quelle que soient les machines qui contiennent les fichiers.
- Les fichiers sont copiés en plusieurs exemplaires pour la fiabilité et permettre des accès simultanés multiples.

HDFS : Architecture (1)



HDFS : Architecture (2)

L'architecture du HDFS est du type Maitre/Esclave , le maitre est appelé **NameNode** qui stocke les métadonnées, l'esclave est **DataNode** qui stocke les données actuelles, HDFS consiste en un seul NameNode et plusieurs DataNodes, en général, l'architecture du HDFS comporte :

- Un NameNode qui contient tous les noms et blocs des fichiers, comme un gros annuaire téléphonique.
- Une autre machine qui est le **secondary NameNode**, une sorte de NameNode de secours, qui enregistre des sauvegardes de l'annuaire à intervalles réguliers.
- Toutes les autres machines sont des DataNode. Elles stockent les blocs du contenu des fichiers.
- D'autres machines qui sont des clients. Ce sont des points d'accès au cluster pour s'y connecter et travailler.

HDFS : Architecture (3)

Les datanodes contiennent des blocs. Les mêmes blocs sont dupliqués (*replication*) sur différents datanodes, en général 3 fois. Cela assure :

• fiabilité des données en cas de panne d'un datanode,

• accès parallèle par différents processus aux mêmes données.

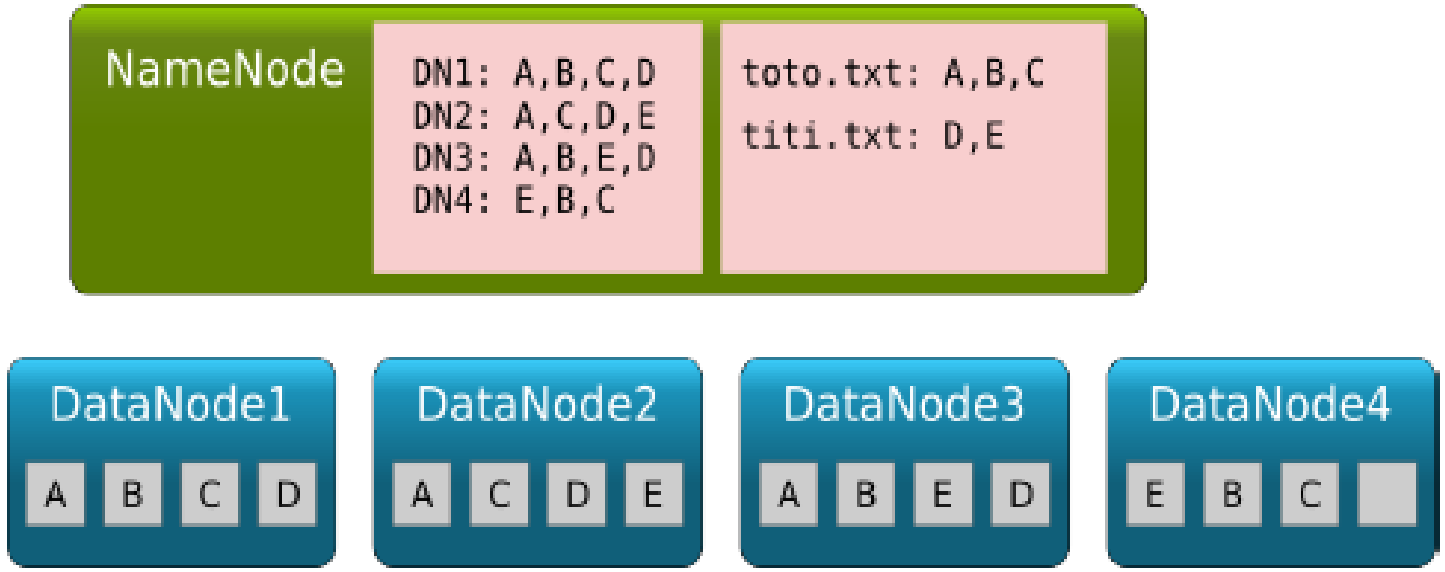
Le namenode sait à la fois :

• sur quels blocs sont contenus les fichiers,

• sur quels datanodes se trouvent les blocs voulus.

On appelle cela les *metadata*.

Inconvénient majeur : panne du namenode = mort de HDFS, c'est pour éviter ça qu'il y a le secondary namenode. Il archive les metadata, par exemple toutes les heures.



Les datanodes contiennent des blocs (A, B, C. . .), le namenode sait où sont les fichiers : quels blocs et sur quels datanodes.

NameNode (1) : Nœud principal

Stocke les méta-données, à savoir le nombre de blocs de données, de répliques et d'autres détails. Ces métadonnées sont disponibles en mémoire dans le maître pour une récupération plus rapide des données. NameNode gère les nœuds esclaves et leur attribue des tâches. Il devrait être déployé sur du matériel fiable, car il s'agit de la pièce maîtresse de HDFS.

Les tâches du NameNode:

- Gérer l'espace de noms du système de fichiers.
- Contrôler l'accès du client aux fichiers.
- Exécuter l'exécution du système de fichiers tel que nommer, fermer, ouvrir des fichiers / répertoires.
- Prendre en compte le facteur de réplication de tous les blocs.

NameNode (2)

Les fichiers présents dans les métadonnées NameNode sont les suivants:

FsImage : «fichier image», contient la totalité de l'espace de noms du système de fichiers et est stocké sous forme de fichier dans le système de fichiers local de NameNode.

EditLogs : Contient toutes les modifications récentes apportées au système de fichiers sur la plus récente FsImage. NameNode reçoit une demande de création / mise à jour / suppression du client. Après cela, cette demande est d'abord enregistrée dans le fichier de modifications.

DataNode : Esclave

DataNode stocke les données réelles dans HDFS. Il effectue des opérations de lecture et d'écriture selon la demande du client. DataNode peut se déployer sur du matériel standard.

Les tâche du DataNode :

- Bloquer la création, la suppression et la réplication de répliques conformément aux instructions de NameNode.
- Gérer le stockage de données du système.
- Envoyer des pulsations au NameNode pour signaler la santé de HDFS. Par défaut, cette fréquence est définie sur 3 secondes.

Secondary NameNode

Lorsque NameNode démarre, il lit d'abord l'état HDFS à partir d'un fichier image, FsImage. Ensuite, il applique les modifications du fichier journal des modifications. NameNode écrit ensuite le nouvel état HDFS dans FsImage.

Ensuite, il commence son fonctionnement normal avec un fichier d'édition vide. Au moment du démarrage, NameNode fusionne FsImage et édite les fichiers afin que le fichier journal d'édition puisse devenir très volumineux au fil du temps. L'un des effets secondaires d'un fichier d'édition plus volumineux est que le prochain redémarrage de NameNode prend plus longtemps.

NameNode secondaire résout ce problème. Secondary NameNode télécharge les fichiers FsImage et EditLogs à partir du NameNode. Et fusionne ensuite EditLogs avec le FsImage (FileSystem Image). Il garde la taille du journal des modifications dans une limite. Il stocke le FsImage modifié dans un stockage persistant. Et nous pouvons l'utiliser en cas d'échec de NameNode.
Secondary NameNode effectue un point de contrôle régulier dans HDFS.

Notion de Block

HDFS divise d’énormes fichiers en petits morceaux appelés blocs (multiple de 64Mo). Ce sont la plus petite unité de données d'un système de fichiers.

La taille par défaut du bloc HDFS est de 128 Mo, ce que nous pouvons configurer en fonction des besoins. Tous les blocs du fichier ont la même taille, à l’exception du dernier, qui peut avoir la même taille ou moins.

Si la taille des données est inférieure à la taille du bloc, celle-ci sera égale à la taille des données. Par exemple, si la taille du fichier est de 129 Mo, 2 blocs seront créés. Un bloc aura une taille par défaut de 128 Mo. L'autre sera de 1 Mo seulement et non de 128 Mo car cela gaspillerait de l'espace.

Hadoop est suffisamment intelligent pour ne pas gaspiller un reste de 127 MB. Donc, il alloue un bloc de 1 Mo uniquement pour des données de 1 Mo. Le principal avantage du stockage de données dans une telle taille est qu’il permet de gagner du temps de recherche du disque.

Gestion de la réplication

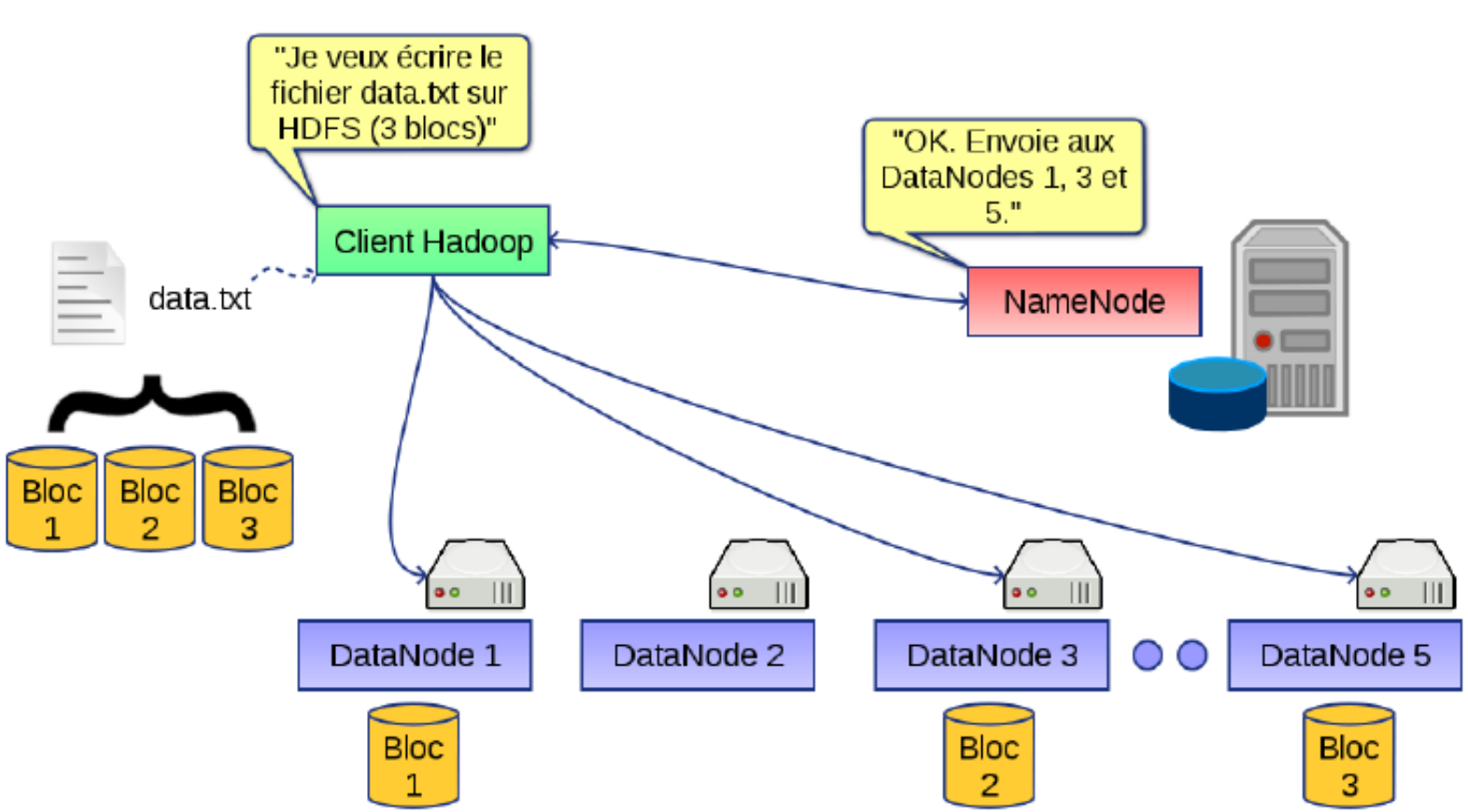
La réplication en bloc offre une tolérance aux pannes. Si une copie n'est pas accessible et corrompue, nous pouvons lire les données d'une autre copie. Le nombre de copies ou de répliques de chaque bloc d'un fichier dans HDFS est un facteur de réplication. Le facteur de réplication par défaut est 3, qui sont à nouveau configurables. Ainsi, chaque bloc est répliqué trois fois et stocké sur différents DataNodes.

Si nous stockons un fichier de 128 Mo dans HDFS en utilisant la configuration par défaut, nous occuperons un espace de 384 Mo (3 * 128 Mo).

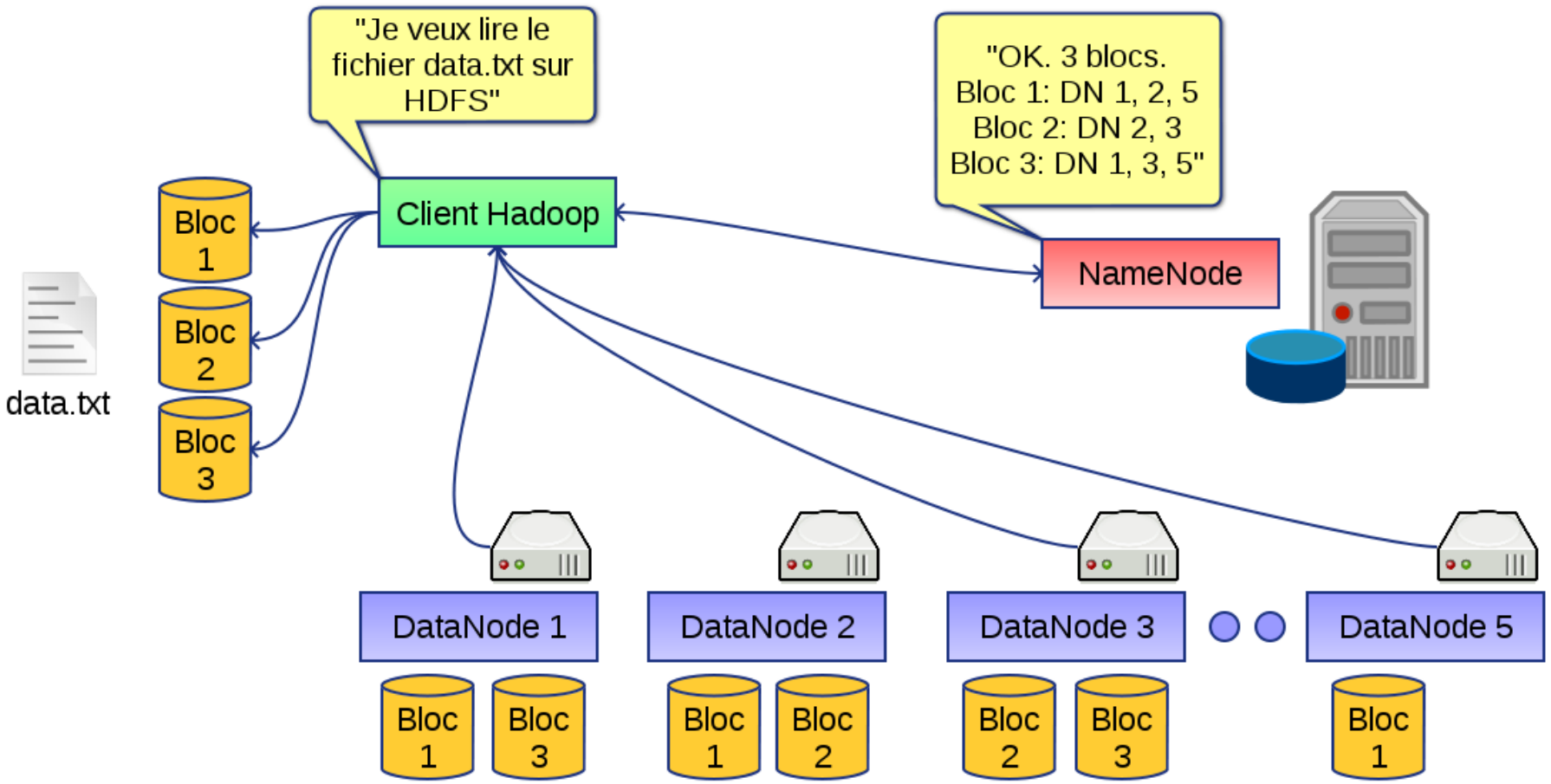
NameNode reçoit périodiquement le rapport du bloc du DataNode afin de gérer le facteur de réplication. Lorsqu'un bloc est sur-répliqué / sous-répliqué, NameNode ajoute ou supprime les réplicas selon les besoins.

HDFS : Écriture d'un fichier

- Le client indique au NameNode qu'il souhaite écrire un bloc.
- Celui-ci lui indique le DataNode à contacter.
- Le client envoie le bloc au Datanode.
- Les DataNodes répliquent le bloc entre eux.
- Le cycle se répète pour le bloc suivant.



HDFS : Lecture d'un fichier



- Le client indique au NameNode qu'il souhaite lire un fichier.
- Celui-ci lui indique sa taille et les différents DataNode contenant les N blocs.
- Le client récupère chacun des blocs à un des DataNodes.
- Si un DataNode est indisponible, le client le demande à un autre.

HDFS : Avantages / Inonvénients

Avantages

- L'ensemble du système de fichier virtuel apparaît comme un disque « unique » : on ne se soucie pas de l'emplacement réel des données.
- Tolérant aux pannes: blocs répliqués.

Inconvénients

- NameNode unique: si problème sur le serveur en question, HDFS est indisponible. Compensé par des architectures serveur active/standby.
- Optimisé pour des lectures concurrentes; sensiblement moins performant pour des écritures concurrentes.

Conclusion

HDFS avec l'aide de NameNode et DataNode :

- Stocke de manière fiable des fichiers très volumineux sur des ordinateurs d'un grand cluster.
- Stocke chaque fichier sous forme de séquence de blocs. La réplication de bloc offre une tolérance aux pannes. Il offre une haute disponibilité, car les données sont hautement disponibles en dépit d'une panne matérielle.
- Lorsqu'une machine ou un matériel plante, nous pouvons accéder aux données d'un autre chemin.