doute l'explosion d'une chaudière située dans ce bâtiment. Cette interprétation n'est possible qu'avec un catalogue des immeubles avec leurs caractéristiques. Avec des millions de capteurs, le passage à l'échelle du traitement des flux envoyés est un défi. Le déploiement des compteurs Linky est un exemple.

LES LIMITES DES ALGÈBRES

Malgré tous ses avantages, les modèles de calcul «à la relationnelle» ne couvrent pas les besoins de l'ensemble des applications, et notamment ceux de deux grands domaines. Dans le premier, on trouve des applications dont les données sont conceptuellement relationnelles, mais dont les performances et certaines contraintes (telle la représentation en ligne des tables) ne sont pas satisfaisantes. C'est le cas des applications nécessitant une représentation des tables par colonnes plus adaptée aux calculs d'agrégats, ou dont la taille des tables, gigantesques, conduit à des performances médiocres de la part des systèmes de gestion de base de données.

Ce domaine est illustré par les applications décisionnelles faisant de gros calculs sur les colonnes et par des applications en astrophysique faisant des calculs sur des tables immenses. Dans les deux cas, même si conceptuellement on manipule des tables, on développe des algorithmes *ad hoc* opérant sur des données massivement distribuées et exploitant des architectures parallèles et des *clusters* de machines.

MapReduce permet une parallélisation des tâches en distribuant les données plutôt que le code

Le second domaine d'applications est illustré, au-delà des volumes de données importants, soit par des données non relationnelles (documents, images…), donc sans connaissance préalable d'une structure, soit par des logiques de traitement qui ne s'expriment pas aisément avec l'algèbre relationnelle. C'est le cas des algorithmes de classification ou d'apprentissage. Là encore, on a recours à une programmation *ad hoc*.

Notons que dans les deux domaines d'applications, le relationnel n'est pas exclu, mais seulement limité à quelques tâches. Par exemple, dans un projet de *machine learning*, la phase de prétraitement peut s'appuyer sur une technologie relationnelle, alors que la phase d'apprentissage proprement dite sera faite par un algorithme approprié.

Cette approche, notée NoSQL, n'est pas nouvelle, des interfaces entre les langages de requêtes et les langages de programmation ont toujours existé pour développer des applications complexes, sous-traitant aux systèmes de gestion de base de données ce qu'ils peuvent faire de mieux et laissant le programmeur réaliser les autres tâches. On la retrouve également dans les systèmes à base de *workflows* où l'on coordonne un ensemble d'activités pouvant être réalisées par des technologies différentes.

Aujourd'hui, ce mode d'ingénierie est toujours pratiqué, avec un regard plus spécifique sur ces tâches. Il s'agit en général d'opérations complexes dont l'optimisation nécessite la mobilisation de ressources de calcul particulières. Ce type de développement est généralement guidé par des patrons (ou *patterns*) de programmation qui intègrent l'invocation de services de données existant et le code de l'utilisateur.

Un de ces *patterns*, connu depuis longtemps en programmation fonctionnelle, est MapReduce (*voir la figure page suivante*). Il permet une parallélisation des tâches à grande échelle en distribuant les données au lieu de distribuer le code. Pour illustrer ce *pattern*, imaginons une opération visant à classifier un sac de billes en fonction de leurs couleurs et à évaluer ensuite la taille de chaque classe et le nombre total de billes. Cette opération peut être réalisée par une seule personne, mais elle sera longue, surtout si le sac de billes est très grand. On recrute alors dix enfants entre qui on partage les billes. En augmentant le parallélisme, on accélère le processus, mais on a à la fin dix classements, chacun produisant des paquets de billes selon leurs couleurs.

Après un travail intermédiaire de regroupement des paquets par couleur, on recrute un nouveau groupe d'enfants et on leur demande de compter le nombre de billes de chaque couleur. À la fin, une seule personne fait la somme des résultats intermédiaires.

Dans ce principe général du modèle de calcul MapReduce, la phase *map* répartit les données (les billes) sur plusieurs serveurs (les enfants), chacun réalisant la même tâche (reconnaissance de la couleur et classement des billes). La phase *reduce* prend les paquets de données intermédiaires et fait la tâche suivante (le comptage). Entre les deux, une «boîte noire» (le *shuffle*) regroupe les paquets par couleur, avant de les acheminer aux serveurs >

> suivants qui feront le *reduce*. Les opérations *map* et *reduce* sont dépendantes de l'application et donc programmées de façon appropriée, alors que le *shuffle* est une opération générique. Ce modèle de calcul permet un gain de temps significatif grâce à sa capacité de mobilisation de ressources à chacune de ses phases d'exécution, mais le prix à payer est au niveau du développement qui nécessite une compétence pointue et au niveau de la mise en œuvre qui dépend d'un grand nombre de paramètres techniques.

Ce modèle est la pierre angulaire de nombreux systèmes de traitement massif des données. Hadoop est l'un des représentants de ces systèmes, combiné avec un système de fichier massivement distribué, de nombreuses extensions ont été réalisées soit au cœur du système pour améliorer ses performances soit au-dessus du système (on parle de surcouches) pour offrir des interfaces de développement plus riches et plus simples.

AU-DELÀ DU CALCUL

Si les opérations de calcul sur les données sont importantes et cristallisent à juste titre l'attention des spécialistes et des programmeurs, la gestion des données ne se limite pas à ces opérations. L'indexation des données et les techniques de hachage, c'est-à-dire la répartition des données sur divers disques, sont cruciales pour la gestion des données, quel que soit le modèle de calcul choisi. Indexer des tables comportant des milliards de lignes ou des documents de plusieurs centaines de millions de pages est une tâche coûteuse. Les techniques mises en œuvre rivalisent d'ingéniosité à la fois dans la façon de construire l'index pour accélérer les accès aux données et dans les performances de cette construction.

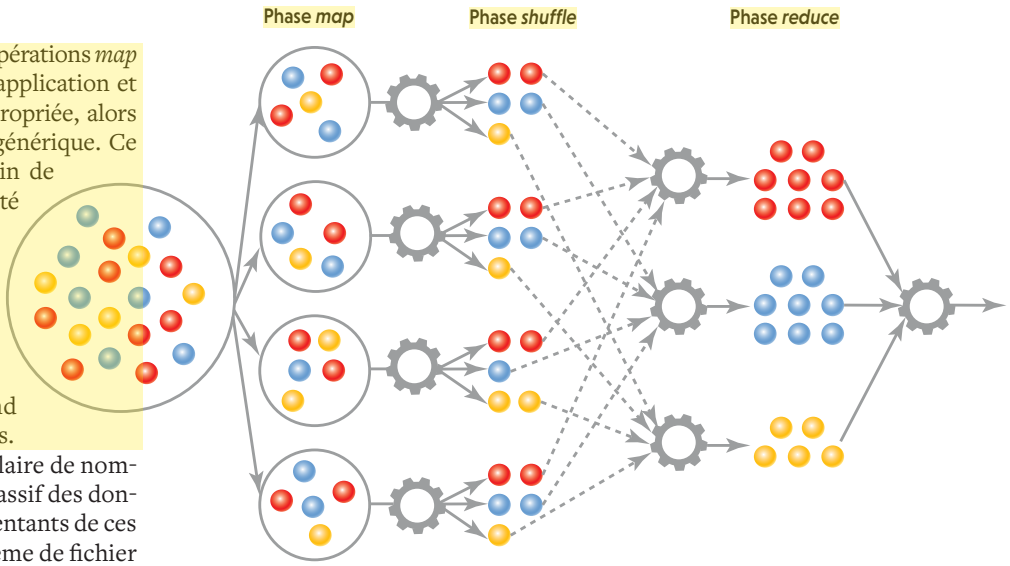
Outre l'indexation et le hachage, la réplication des données est aussi un bon moyen d'augmenter la disponibilité de l'information en multipliant les exemplaires et les points d'accès. La réplication est très importante dans un système distribué afin de réduire les transferts de données, souvent coûteux, sur le réseau. Cependant, les avantages de l'indexation et de la réplication sont vite anéantis quand les données sont fréquemment mises à jour, ce qui n'est heureusement pas le cas dans de nombreuses applications des *big data*.

La protection des données reste un sujet sensible lorsqu'il s'agit de données personnelles, de santé ou concernant des processus industriels. De nombreux systèmes intégrés de gestion de données sont dotés de mécanismes d'anonymisation (*voir L'art de préserver l'anonymat, par T. Allard, page 98*), de cryptage, ou de restriction d'accès à certaines données. Des hiérarchies de droits sont souvent associées à des hiérarchies d'utilisateurs contrôlant ainsi l'accès aux données sensibles. Les algorithmes associés à ces garde-fous sont souvent complexes.

Dans le registre des risques, ajoutons que la complexité des algorithmes mis en œuvre dans l'exploitation des *big data* peut entraîner des effets de bord qui, si l'on n'y prend pas garde, peuvent poser des problèmes d'éthique, violer la vie privée, réduire la liberté et avoir un impact fortement intrusif dans la vie des personnes. Ce sont des questions essentielles mises en débat récemment (*voir l'entretien avec N. Boujemaa, page 104*).

La sensibilité des données n'est pas seulement liée à leur usage, elle peut être liée à la technologie. Les erreurs de matériel ou logicielles ne sont pas rares. Ces incidents ne doivent pas mettre le système de données dans un état incohérent, avec des risques de perte d'informations. Des mécanismes de fiabilisation sont développés, notamment au niveau des systèmes de gestion de base de données pour protéger ces systèmes et permettre un redémarrage à chaud ou à froid, sans perte d'informations et sans incohérence sur les données globales. Ces algorithmes de protection des données à tous les niveaux occupent une place stratégique. Ils sont complexes et leur mise en œuvre n'est pas sans incidence sur les coûts de calcul.

En fin de compte, les algorithmes destinés à la gestion des *big data* constituent une vaste faune, toujours plus diversifiée et riche à mesure des développements. C'est que, pour filer la métaphore, leur monde, celui des *big data*, ne cesse de croître. D'ailleurs, la comparaison avec le pétrole trouve ici ses limites. Les ressources en cette énergie fossile sont inéluctablement appelées à se tarir, alors que celles en données sont en augmentation permanente. Et une donnée peut être utilisée autant de fois que nécessaire sans se consumer. ■



MapReduce accélère le processus de traitement des données grâce au parallélisme, c'est-à-dire à une distribution des tâches. Ici, elle consiste en une répartition des données pour les classer. D'abord, le paquet initial est divisé en quatre parties *(à gauche)*, chacune étant ensuite classée *(au centre)*. Les résultats partiels sont enfin réunis en un seul *(à droite)*.

BIBLIOGRAPHIE

M. BOUZEGHOUB ET R. MOSSERI (DIR.), ***Les Big Data à découvert,*** **CNRS Éditions, 2017.**

S. ABITEBOUL, «Sciences des données: de la logique du premier ordre à la Toile», Leçon inaugurale de la Chaire d'Informatique et sciences numériques au Collège de France, 2012.