IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Reçu le 24 juillet 2023, accepté le 5 août 2023, date de publication le 9 août 2023, date de la version actuelle le 16 août 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3303810

SURVEY

Enquête sur le streaming de données en réseau avec Apache Kafka

THEOFANIS P. RAPTIS ET ANDREA PASSARELLA
Institut d'informatique et de télématique, Conseil national de la recherche, 56124 Pise, Italie
Auteur correspondant : Theofanis P. Raptis (theofanis.raptis@iit.cnr.it)

Ce travail a été soutenu par le programme de recherche et d'innovation Horizon 2020 de l'Union européenne, MARVEL, dans le cadre de la subvention 957337.

RÉSUMÉ Apache Kafka est devenu une solution populaire pour la gestion des flux de données en réseau dans une variété d'applications, de l'industrie à l'usage général. Cet article passe systématiquement en revue la littérature de recherche dans ce domaine en la classant soigneusement dans des macro-domaines clés, à savoir les algorithmes, les réseaux, les données, les systèmes cyber-physiques et la sécurité. Grâce à cette classification méticuleuse, le document vise à identifier et à analyser les aspects d'optimisation pertinents pour chaque domaine, en s'appuyant sur des applications pratiques comme base d'analyse. À cet égard, le document synthétise et consolide les connaissances existantes, ce qui permet aux chercheurs d'économiser du temps et des efforts précieux dans la recherche d'informations pertinentes dans de multiples sources. Les avantages tangibles de ce document d'étude sont les suivants : fournir une base de connaissances consolidée sur les sujets Apache Kafka à forte intensité de recherche, mettre en évidence les idées pratiques et les nouvelles approches, mettre en évidence les applications interdomaines, identifier les défis de recherche connexes et servir de référence fiable pour la communauté Apache Kafka.

TERMES DE L'INDEX Algorithmes, cyber-physique, données, Internet des objets, réseaux, pub-sub, sécurité, traitement des flux.

I. INTRODUCTION

Le flux de données en réseau est un paradigme méthodologique essentiel qui est devenu de plus en plus important dans le paysage numérique actuel, qui évolue rapidement. Comme le montre la figure 1, le flux de travail comprend plusieurs étapes pour transmettre et traiter des données en temps réel : Les données en temps réel sont générées par diverses sources en réseau, telles que des capteurs, des appareils ou des applications logicielles, et transmises sur un réseau informatique tel qu'Internet ou un réseau privé. Les données en temps réel sont ensuite publiées dans un courtier de messages, qui agit comme une plaque tournante centrale qui reçoit et distribue les données à de multiples abonnés en temps réel. Ce processus est connu sous le nom de publication/souscription (pub/sub) et permet aux données d'être partagées efficacement entre plusieurs applications ou systèmes de traitement. Les données sont ensuite traitées en temps réel à l'aide de la technologie de traitement des flux. Le traitement de flux consiste à appliquer des algorithmes ou des règles au flux de données lorsqu'il circule dans le système, ce qui permet une analyse, une agrégation, un filtrage ou une transformation immédiats des données. Les données traitées sont ensuite restituées sous la forme d'un flux d'informations

Le rédacteur en chef adjoint chargé de coordonner la révision de ce manuscrit et d'en approuver la publication est Guangjie Han .

dans un format structuré, souvent en temps réel. Ce flux de sortie peut être consommé à des fins d'analyse ou de prise de décision par de multiples applications, tableaux de bord ou outils de visualisation qui utilisent les données traitées à des fins différentes, telles que la génération d'alertes, la mise à jour de bases de données ou le déclenchement d'actions automatisées.

Avec les quantités massives de données générées chaque jour, il est devenu crucial pour les organisations de traiter et d'analyser ces données en temps réel. La diffusion de données en réseau permet aux organisations de fournir un moyen efficace de transmettre rapidement des données provenant de diverses sources à un emplacement central pour traitement et analyse. Les applications de la diffusion en continu de données en réseau sont variées et étendues. Par exemple, elle peut être utilisée pour la surveillance en temps réel du trafic et des transactions financières sur les réseaux des villes intelligentes [1], ainsi que pour l'analyse des données des capteurs dans l'industrie manufacturière afin de prévoir les pannes d'équipement [2] et pour la détection et l'actionnement sur de vastes champs agricoles intelligents [3]. En fournissant des informations en temps réel sur les opérations commerciales, le flux de données en réseau permet aux organisations de prendre des décisions éclairées plus rapidement, améliorant ainsi l'efficacité et la compétitivité globales. L'un des principaux avantages de la diffusion en continu de données en réseau est sa capacité à identifier les problèmes potentiels.

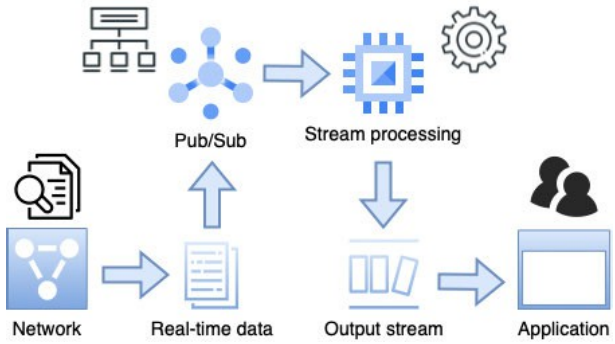

FIGURE 1. Flux de données en réseau.

les problèmes avant qu'ils ne s'aggravent, réduisant ainsi les temps d'arrêt et améliorant la productivité.

La conception d'un système de flux de données en réseau utilisant les technologies pub- lish/subscribe (pub/sub) est une approche populaire pour traiter et analyser de grands volumes de données en temps réel (figure 1). Pub/sub est un paradigme de messagerie qui permet aux éditeurs d'envoyer des messages aux abonnés qui ont exprimé leur intérêt pour des sujets spécifiques. L'utilisation des technologies pub/sub dans les flux de données en réseau offre plusieurs avantages [4]. Tout d'abord, il s'agit d'une solution flexible et évolutive qui peut s'adapter à l'évolution des volumes et des sources de données. Ensuite, elle permet un traitement efficace des données en les répartissant sur plusieurs nœuds du réseau. Enfin, il permet une intégration transparente avec d'autres systèmes et services, ce qui facilite l'exploitation des données pour divers cas d'utilisation. Lors de la conception d'un système de streaming de données en réseau basé sur pub/sub, il est essentiel de prendre en compte des facteurs tels que l'extensibilité, la tolérance aux pannes et la sécurité des données. Il est également important de sélectionner les technologies pub/sub appropriées et de les configurer correctement pour garantir des performances et une fiabilité optimales.

Apache Kafka [5] est une plateforme de streaming distribué à code source ouvert très populaire, largement utilisée pour créer des applications de streaming de données en réseau, telles que LinkedIn [6]. Elle fournit un système de messagerie pub/sub qui permet de traiter les données en temps réel et de les distribuer sur plusieurs nœuds du réseau. L'utilisation d'Apache Kafka comme système pub/sub pour le streaming de données en réseau offre une solution hautement évolutive et tolérante aux pannes qui a le potentiel de traiter facilement des quantités massives de données. Cependant, lors de l'utilisation d'Apache Kafka en tant que système pub/sub pour le streaming de données en réseau, il est important de prendre en compte des facteurs d'optimisation tels que la sérialisation des données, la taille des messages et le partitionnement, et donc de configurer Apache Kafka correctement afin de garantir les performances et la fiabilité souhaitées. Par conséquent, de nombreuses recherches ont été menées sur divers aspects d'Apache Kafka afin de mieux comprendre ses capacités, ses limites et ses applications potentielles. En tant que plateforme open-source, elle est en constante évolution et de nouvelles fonctionnalités et améliorations sont régulièrement ajoutées. Les chercheurs ont donc la possibilité d'explorer et d'expérimenter la plateforme et de développer de nouveaux cas d'utilisation et de nouvelles applications. L'effervescence de la recherche sur Apache Kafka reflète son importance croissante en tant que plateforme de flux de données et sa capacité à

pour "révolutionner" constamment la manière dont les organisations traitent et analysent les données en temps réel.

Dans cet article, nous passons en revue la littérature de recherche sur Apache Kafka pour le streaming de données en réseau afin de fournir une vue d'ensemble complète de l'état actuel des connaissances sur la plateforme. De nombreuses recherches ont été menées sur divers aspects d'Apache Kafka, notamment son architecture, ses performances, son évolutivité et sa sécurité. Cependant, à notre connaissance, aucune étude littéraire n'a abordé l'organisation des connaissances sur le sujet de manière synthétique. Pour combler cette lacune, nous synthétisons et résumons la littérature existante sur Apache Kafka, afin de donner un aperçu de l'état actuel des connaissances et d'identifier les lacunes et les possibilités de recherche future. Ce document peut constituer une ressource précieuse pour les praticiens qui cherchent à tirer parti d'Apache Kafka pour leurs besoins en matière de flux de données, en fournissant un aperçu complet des capacités de la plateforme, de ses limites et des cas d'utilisation potentiels. Dans l'ensemble, notre document apporte les contributions suivantes :

1) Une classification de la littérature de recherche en représenta
   macro-domaines (algorithmes, réseaux, données, cyber-physique et sécurité) et identification des aspects d'optimisation correspondants, sur la base d'applications pratiques
2) Exploration des fondements algorithmiques d'Apache Kafka, y compris ses aspects combinatoires et de fiabilité
3) Couverture du domaine de l'optimisation des infrastructures en réseau, en examinant comment Kafka peut améliorer les performances et l'évolutivité des systèmes distribués.
4) Discussion sur la manipulation et le traitement des données, en soulignant comment Apache Kafka peut être utilisé pour le streaming de données en temps réel, la mise en file d'attente de messages et l'informatique basée sur le ML.
5) Étude de la tendance émergente de la convergence cyber-physique et du rôle d'Apache Kafka dans l'intégration des systèmes physiques avec des applications axées sur les données, dans le contexte de l'internet des objets, des véhicules, de la mobilité et des cas d'utilisation environnementaux.
6) Couverture des considérations de sécurité liées à Apache Kafka et de la manière dont la plateforme peut être utilisée en toute sécurité dans les environnements d'entreprise.
7) Identification des défis de recherche ouverts sélectionnés pour le streaming en réseau avec Apache Kafka.

Après avoir évalué l'état de l'art dans la section II et, comme le montre la figure 2, présenté les bases d'Apache Kafka dans la section III, nous présentons une étude approfondie d'Apache Kafka, en analysant les dernières recherches et évolutions dans plusieurs domaines critiques liés à Kafka. Le document est structuré en plusieurs sections, chacune se concentrant sur des sujets spécifiques liés à Apache Kafka. La section IV se penche sur les fondements algorithmiques d'Apache Kafka, en explorant son architecture et ses principes de conception. La section V traite de l'optimisation des infrastructures en réseau et examine comment Kafka peut améliorer les performances et l'évolutivité des systèmes distribués. La section VI explore la manipulation et le traitement des données et examine comment Apache Kafka peut être utilisé pour le streaming de données en temps réel, la mise en file d'attente de messages et l'informatique basée sur le ML. Section VII

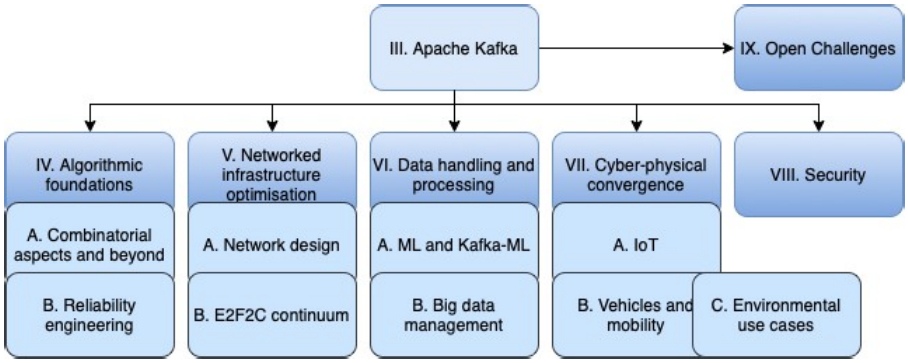

FIGURE 2. Feuille de route du document.

traite de la tendance émergente de la convergence cyber-physique, en soulignant le rôle d'Apache Kafka dans l'intégration des systèmes physiques avec les applications basées sur les données. La section VIII se concentre sur les considérations de sécurité liées à Apache Kafka, en explorant comment la plateforme peut être utilisée en toute sécurité dans les environnements d'entreprise. Enfin, dans la section IX, nous identifions quelques défis de recherche ouverts.

II. MÉTHODOLOGIE ET APERÇU DE LA LITTÉRATURE

Ce document d'étude suit une méthodologie d'analyse systématique de la littérature afin de fournir une analyse complète de la recherche existante sur les sujets liés à Apache Kafka. Le processus d'analyse systématique a comporté plusieurs étapes clés :

1) Formulation des questions de recherche : Nous avons défini des questions de recherche claires pour guider notre recherche documentaire et notre analyse. Ces questions portent sur les aspects clés d'Apache Kafka, notamment son architecture, ses performances, son évolutivité, sa fiabilité, sa sécurité et son intégration avec d'autres systèmes.
2) Stratégie de recherche : Nous avons élaboré une stratégie de recherche complète pour identifier les études pertinentes. La recherche a été effectuée dans diverses bases de données universitaires, notamment IEEE Xplore, ACM Digital Library et Google Scholar. Nous avons également consulté des comptes rendus de conférences, des rapports industriels et de la documentation technique afin d'assurer une large couverture de la littérature. Nous avons combiné et utilisé une série de mots-clés afin d'obtenir une couverture maximale du domaine, tels que Kafka, streaming, data, service, network.
3) Sélection des études : Nous avons appliqué des critères d'inclusion et d'exclusion prédéfinis pour sélectionner les études qui répondaient à nos objectifs de recherche. Le principal critère de recherche pour les publications connexes est la présence d'une utilisation ou d'une avancée notable d'Apache Kafka et son applicabilité dans le contexte de la diffusion en continu en réseau. Le nombre d'articles trouvés était supérieur à 90. Toutefois, sur la base de notre propre jugement et en utilisant comme critère d'exclusion l'absence d'une contribution solide d'Apache Kafka au cœur d'un travail donné, le nombre final d'articles pris en compte s'est finalement élevé à 70.

4) Extraction et analyse des données : Nous avons extrait les données pertinentes des études sélectionnées, y compris les informations sur les objectifs, la méthodologie, les résultats et les contributions de l'étude. Nous avons analysé les données extraites pour identifier les thèmes communs, les tendances émergentes et les lacunes de la recherche dans le domaine d'Apache Kafka.
5) Évaluation de la qualité : Nous avons évalué la qualité des études sélectionnées à l'aide de critères établis tels que la pertinence de la question de recherche, la rigueur de la méthodologie et la validité des résultats.
6) Synthèse et rapport : Nous avons synthétisé les résultats des études sélectionnées et les avons organisés par thème. Les résultats sont présentés de manière structurée et cohérente dans le document d'enquête, qui donne un aperçu de l'état actuel de la recherche sur les sujets liés à Apache Kafka.

Bien qu'Apache Kafka soit déjà une plateforme commercialement populaire, il existe, à notre connaissance, un nombre très limité d'articles anciens qui font état de certaines avancées systématiques de la recherche dans ce domaine de manière exhaustive. C'est pourquoi nous avons étendu notre recherche aux articles qui explorent les avantages et les inconvénients d'Apache Kafka dans son ensemble. Nous dressons la liste de ces articles dans le tableau 1 et nous les comparons à notre contribution. Il convient de noter qu'aucun des articles identifiés n'est une étude pure ; il s'agit plutôt de contributions techniques qui offrent toutefois un aperçu détaillé de leur domaine de recherche de référence. Plus précisément, nous avons effectué la comparaison en fonction des parties thématiques fondamentales du présent document : (i) fondements algorithmiques,
(ii) l'infrastructure en réseau, (iii) le traitement des données, (iv) les systèmes cyber-physiques et (v) la sécurité. Comme le montre le tableau 1, aucun document ne couvre toutes les parties thématiques. De plus, les articles précédents ayant été publiés entre 2015 et 2021, l'article actuel couvre naturellement une perspective plus actuelle du sujet. Enfin et surtout, à notre connaissance, le présent document décrit systématiquement, pour la première fois dans l'état de l'art, les aspects cyber-physiques et sécuritaires de la littérature.

III. APACHE KAFKA

Apache Kafka est une plateforme de streaming distribuée conçue pour traiter des quantités massives de données en temps réel. Au cœur de cette plateforme, un cluster Apache Kafka fournit une

TABLEAU 1. Comparaison avec les documents antérieurs portant sur les éléments d'Apache Kafka (2015-2023).

Reference	Year	Algorithmic foundations	Networked infrastructure	Data handling	Cyber-physical	Security
current paper	2023	✓	✓	✓	✓	✓
[7]	2021	✗	✓	✓	✗	✗
[8]	2021	✗	✓	✗	✗	✗
[9]	2019	✓	✓	✓	✗	✗
[10]	2018	✗	✗	✓	✗	✗
[11]	2015	✗	✗	✓	✗	✗

(Fig. 3), et un système de messagerie pub/sub qui permet aux producteurs de publier des données dans les rubriques Kafka et aux consommateurs de s'abonner à ces rubriques et de recevoir les données au fur et à mesure qu'elles arrivent. Dans Kafka, les producteurs sont chargés de publier des données dans les rubriques Kafka. Ils peuvent publier des données dans n'importe quel format, qu'il s'agisse de texte, de données binaires ou de JSON. Lorsqu'un producteur publie des données vers un sujet Kafka, il envoie un message comprenant une clé et une valeur. La clé sert à identifier le message et peut être utilisée pour le partitionnement et l'indexation. La valeur contient les données réelles.

Les sujets Apache Kafka sont des catégories logiques ou des flux de données. Ils sont créés par les administrateurs et peuvent avoir plusieurs producteurs et consommateurs. Les sujets peuvent être partitionnés, ce qui permet un traitement parallèle des messages et une meilleure évolutivité. Lorsqu'un message est publié dans un thème, il est ajouté à la fin du journal du thème. Les partitions dans Kafka sont l'unité de base du parallélisme. Chaque partition est une séquence de messages stockés sur un seul nœud de broker. Lorsqu'un message est publié dans une partition, il se voit attribuer un décalage séquentiel qui représente sa position dans la partition. Les consommateurs peuvent lire les messages d'une partition en parallèle, ce qui permet un débit élevé et une grande évolutivité. Dans Kafka, les répliques sont des copies des partitions qui sont stockées sur plusieurs nœuds de courtier. La réplication offre une tolérance aux pannes et garantit que les données ne sont pas perdues en cas de défaillance d'un courtier. Kafka prend en charge des facteurs de réplication configurables, qui spécifient le nombre de répliques à créer pour chaque partition.

Les consommateurs d'Apache Kafka sont chargés de s'abonner aux sujets Kafka et de lire les messages qu'ils contiennent. Ils peuvent lire des messages provenant d'une ou de plusieurs partitions et peuvent traiter les messages en parallèle pour atteindre un débit élevé. Les consommateurs peuvent également se regrouper pour former des groupes de consommateurs, ce qui permet l'équilibrage de la charge et le basculement. Kafka prend en charge les modèles de consommation basés sur la poussée et la traction. Dans le modèle "push", Kafka envoie les messages aux consommateurs dès qu'ils sont disponibles. Dans le modèle "pull", les consommateurs demandent des messages à Kafka et les reçoivent par lots. Kafka prend également en charge le traitement en flux. Le traitement par flux consiste à traiter les données en temps réel dès leur arrivée dans Kafka, plutôt que de les stocker dans une base de données pour un traitement par lots ultérieur. Kafka Streams est une bibliothèque Java qui fournit une interface de programmation d'applications (API) de haut niveau pour construire des applications de traitement de flux au-dessus de Kafka.

La figure 3 illustre un exemple de cluster Apache Kafka avec quatre courtiers, b_1 , b_2 , b_3 , et b_4 , et deux sujets, τ_1 et τ_2 , chacun avec plusieurs partitions. En outre, il y a deux producteurs de données, p_1 et p_2 , et quatre consommateurs de données, c_1 , c_2 , c_3 , et c_4 , qui peuvent s'abonner à la base de données.

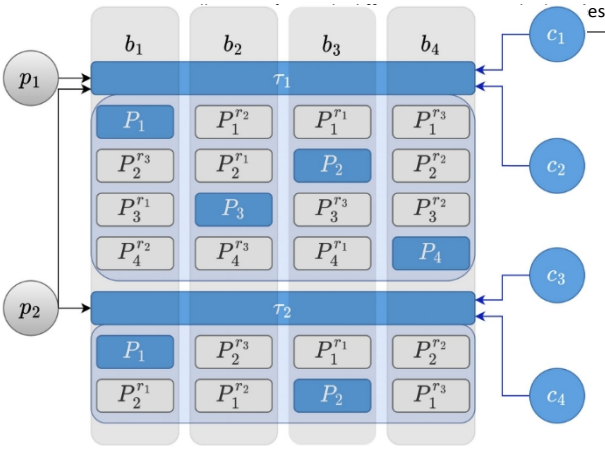

FIGURE 3. Exemple de cluster Apache Kafka composé de quatre courtiers $_{b1, b2, b3, b4}$ pour deux sujets donnés $_{\tau1, \tau2}$, deux producteurs de données $_{p1, p2}$, quatre consommateurs de données $_{c1, c2, c3, c4}$, et un facteur de réplication r = 3. Les partitions leaders sont en bleu, les répliques en gris. Les partitions sont généralement attribuées aux courtiers par le biais d'un algorithme d'attribution de partitionnement par thème.

et consomment des données provenant des thèmes. Les partitions sont utilisées pour diviser un thème en morceaux de données plus petits et plus faciles à gérer, qui peuvent être distribués par plusieurs courtiers. Chaque partition est répliquée plusieurs fois, avec un facteur de réplication de trois dans ce cas, afin de garantir la tolérance aux pannes et la redondance des données. La partition de couleur bleue est la partition leader, qui est responsable de toutes les opérations de lecture et d'écriture pour une partition donnée. Les répliques de couleur grise sont des sauvegardes de la partition leader, qui prennent le relais en cas de défaillance de cette dernière. Lorsqu'un producteur de données, tel que p_1 ou p_2 , envoie un message à un sujet Kafka, le message est d'abord reçu par le courtier Kafka qui est la partition leader pour la partition à laquelle le message est envoyé. La partition leader écrit alors le message sur son disque local et envoie des copies du message aux autres répliques de cette partition. Une fois que les répliques ont accusé réception du message, la partition leader renvoie un accusé de réception au producteur. Les consommateurs de données, tels que c_1 , c_2 , c_3 , et c_4 , peuvent s'abonner à un ou plusieurs thèmes Kafka et consommer des messages provenant des partitions qui leur sont attribuées. Lorsqu'un consommateur rejoint un thème, il se voit attribuer une ou plusieurs partitions à consommer, et chaque groupe de consommateurs est assuré de recevoir tous les messages de chaque partition qui lui est attribuée. En résumé, Apache Kafka décompose les sujets en partitions et les réplique sur plusieurs courtiers, ce qui permet la tolérance aux pannes et la redondance des données. Les producteurs de données envoient des messages à la partition leader d'une partition, qui distribue ensuite les messages aux répliques. Les consommateurs de données peuvent s'abonner à un ou plusieurs sujets Kafka et consommer les messages

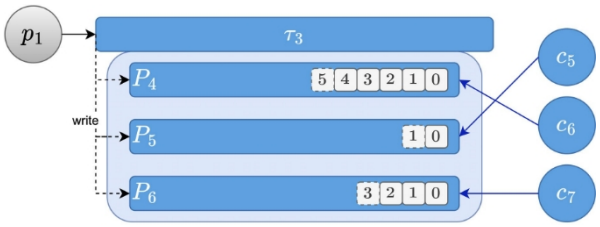

FIGURE 4. Exemple d'un processus d'écriture/lecture Apache Kafka d'un sujet donné $_{\tau 3}$, d'un producteur de données $_{p3}$ et de trois données consommateurs $_{c5, c6, c7}$.

à partir des partitions qui leur sont attribuées. Cette architecture permet un transfert de données en temps réel efficace et évolutif entre plusieurs applications et services.

La figure 4 illustre ce qui se passe lorsqu'un producteur génère un message. Dans l'image, nous avons un sujet Kafka τ_3 , un producteur de données p_1 , et trois consommateurs de données c_5 , c_6 , c_7 . Le sujet est divisé en trois partitions P_4 , P_5 , P_6 pour des raisons d'évolutivité et de performance. Examinons tout d'abord le processus d'écriture. Le producteur de données envoie un message au courtier Apache Kafka en spécifiant le nom du sujet et de la partition. À l'exception de sa valeur, le message contient une clé numérotée, et le courtier ajoute le message à la fin du journal de la partition. Chaque message se voit attribuer un décalage unique dans la partition qui représente sa position dans la partition. Ce décalage est utilisé par les consommateurs pour suivre leur progression et s'assurer qu'ils ne manquent aucun message.

Examinons maintenant le processus de lecture. Chaque consommateur de données s'abonne au thème et à l'une des partitions qui l'intéressent. Lorsqu'un consommateur s'abonne pour la première fois, il spécifie le décalage à partir duquel il souhaite commencer à lire les messages. Le consommateur peut choisir de commencer à lire à partir du premier décalage disponible, ce qui signifie qu'il lira tous les messages depuis le début de la partition, ou à partir d'un décalage spécifique, ce qui signifie qu'il commencera à lire à partir de cette position dans le journal de la partition. Au fur et à mesure que les messages sont écrits dans les partitions, ils deviennent disponibles pour la consommation. Dans l'exemple, les consommateurs utilisent un modèle basé sur l'extraction pour lire les messages des partitions. Le consommateur envoie une requête au courtier pour demander un lot d e messages à partir de sa position de décalage actuelle. Le courtier répond par un lot de messages, que le consommateur traite un par un. Une fois qu'un message est traité, le consommateur met à jour sa position de décalage par rapport au message suivant dans la partition.

Dans notre exemple, nous avons trois consommateurs qui lisent le même sujet, chacun lisant une partition différente. Chaque consommateur conserve sa propre position de décalage et lit les messages indépendamment. Cela signifie que les messages peuvent être traités en parallèle, ce qui permet d'obtenir un débit élevé et une grande évolutivité. Il est important de noter qu'Apache Kafka prend également en charge les groupes de consommateurs, qui permettent à plusieurs consommateurs de travailler ensemble pour lire une partition. Dans un groupe de consommateurs, chaque consommateur se voit attribuer un sous-ensemble d'offsets de la partition à lire. Cela permet de s'assurer que chaque message n'est traité qu'une seule fois, même s'il y a plusieurs consommateurs

dans le groupe. Dans cet exemple, nous avons également supposé qu'il n'y avait pas de répliques de la partition. Cependant, dans un environnement de production, il est courant d'avoir plusieurs répliques de chaque partition pour la tolérance aux pannes et la haute disponibilité, comme le montre la figure 3. Les répliques sont simplement des copies du journal de la partition qui sont stockées sur différents nœuds de courtier. Lorsqu'un courtier tombe en panne, l'une des répliques peut prendre le relais et continuer à répondre aux demandes.

IV. FONDEMENTS ALGORITHMIQUES

Les fondements algorithmiques d'Apache Kafka (tableau 2) sont essentiels pour parvenir à une optimisation combinatoire du processus de partitionnement des sujets et pour garantir la fiabilité des applications de traitement des données et de diffusion en continu à grande échelle. Ces fondements reposent sur des principes essentiels qui sous-tendent la conception et la fonctionnalité de la plateforme Kafka. Dans cette sous-section, nous examinerons les composants clés de l 'architecture Kafka, notamment son système de messagerie distribuée, son partitionnement et son organisation thématique des données, ainsi que ses mécanismes de réplication et de tolérance aux pannes. Nous explorerons également les principes de conception fondamentaux qui ont guidé le développement de Kafka, tels que l'accent mis sur la simplicité, l'évolutivité et la performance. En comprenant les fondements algorithmiques de Kafka, les chercheurs et les praticiens peuvent optimiser l'utilisation de ses capacités pour répondre aux besoins de gestion de données complexes des applications modernes tout en garantissant la fiabilité de l'ingénierie.

A. ASPECTS COMBINATOIRES ET AU-DELÀ

Les aspects combinatoires d'Apache Kafka constituent une part importante de ses fondements algorithmiques, fournissant des approches puissantes pour l'optimisation du traitement des données à grande échelle et de la diffusion en continu. L'optimisation combinatoire consiste à trouver les meilleures solutions à partir d'un ensemble fini de possibilités. Dans le contexte de Kafka, il s'agit de maximiser le débit des flux de données, de réduire la latence et de minimiser l'utilisation des ressources. La littérature offre une quantité considérable de techniques d'optimisation combinatoire utilisées dans Apache Kafka, telles que le traitement par lots, la compression et la mise en lots. En comprenant les aspects combinatoires et au-delà d'Apache Kafka, les chercheurs et les praticiens peuvent explorer de nouvelles façons d'optimiser le traitement des données et le streaming pour répondre aux demandes toujours croissantes des applications modernes.

Une première catégorie d'optimisations se concentre sur la modélisation et la conception du transfert de données. Malgré l'importance et la croissance de la communauté d'utilisateurs, il subsiste une lacune importante dans les approches de modélisation formelle qui peuvent être utilisées pour raisonner sur le comportement des producteurs et des consommateurs dans les systèmes basés sur Kafka. L'un des principaux défis de la formalisation du modèle de transfert de données de Kafka est qu'il implique de multiples couches d'abstraction, des protocoles de réseau de bas niveau à la sémantique de traitement des messages de haut niveau. En outre, les producteurs et les consommateurs peuvent interagir avec Kafka de nombreuses manières différentes, en fonction de facteurs tels que les stratégies de partitionnement, les paramètres de réplication et les configurations de la bibliothèque client. Tous ces facteurs font qu'il est difficile de développer des modèles généraux qui capturent avec précision le comportement des systèmes Kafka du monde réel.

TABLEAU 2. Fondements algorithmiques d'Apache Kafka.

Article	Modelling consideration	Methodology
Combinatorial aspects		
[12]	Data transfer	Formal methods
[13]		Queueing theory
[14]	Partitioning	Integer programming, heuristics
[15]		Bin packing, R-score
[16]	Consensus	Control theory
[17]	Data starvation	Load shedding
[18]	Service overhead	Simulation modelling
Reliability engineering		
[19]	Fault tolerance	Checkpoint interval values optimisation
[20]		Cooperative clustering
[21]		Latency engineering
[22]		Architectural configurations
[23]	Reliability estimation	
[24]	Reliability prediction	Neural networks
[25]	Reliability assesment	Testing tool

Suivant ce raisonnement, dans [12], la communication dans Apache Kafka entre les producteurs et les consommateurs est modélisée à l'aide de méthodes formelles. Certaines caractéristiques du système sont vérifiées à l'aide des outils de test du modèle. Les résultats de la vérification démontrent que le modèle de transfert de données d'Apache Kafka respecte ses spécifications, ce qui permet de conclure que le système est digne de confiance. Dans [13], afin d e prévoir les mesures de performance des services en nuage Apache Kafka, les auteurs analysent les caractéristiques structurelles d'Apache Kafka et proposent un modèle de transfert de données inspiré des méthodes de file d'attente. Le nombre de courtiers dans le cluster Apache Kafka, le nombre de partitions dans un sujet et la taille du lot de données sont les entrées de configuration initiales de cette approche. L'effet de facteurs de configuration spécifiques sur les mesures de performance, telles que le rendement du producteur, la charge utile relative et les frais généraux, ainsi que les changements dans l'utilisation du stockage sur disque au fil du temps, peuvent être déterminés par les utilisateurs à l'aide de ce modèle. La théorie des files d'attente est utilisée pour évaluer le délai de bout en bout des messages.

Une deuxième catégorie importante de travaux porte sur la manière de résoudre les problèmes liés au partitionnement des sujets ou aux affectations partition-consommateur. Bien qu'Apache Kafka comprenne certaines optimisations prêtes à l'emploi, les auteurs de [14] soulignent qu'il ne spécifie pas explicitement comment chaque sujet doit être partitionné afin d'être distribué efficacement. À cet égard, ils commencent par simuler la façon dont Apache Kafka partitionne les sujets pour un sujet spécifique. Ils posent ensuite le problème d'optimisation consistant à déterminer le nombre requis de partitions et démontrent qu'il est difficile à calculer (il peut être formulé sous la forme d'un programme en nombres entiers). Les auteurs proposent deux méthodes simples mais efficaces pour résoudre le problème : la première vise à maximiser le nombre de courtiers utilisés dans le cluster, tandis que la seconde le minimise. Les auteurs de [15] utilisent une variante du problème du bin packing pour abstraire le défi de trouver le nombre nécessaire de consommateurs et les affectations partition-consommateur. Ils proposent des mesures indicatives qui prennent en compte les dépenses de rééquilibrage, et introduisent et évaluent une variété de méthodes par rapport aux stratégies connues pour le problème du bin packing dans ce contexte.

Les travaux restants sur les aspects combinatoires avec Apache Kafka se concentrent sur une variété de problèmes différents. Nous les regroupons dans ce paragraphe et en donnons une brève description ; ils portent principalement sur (i) les algorithmes de consensus basés sur Kafka, (ii) la modélisation de la pénurie de données et (iii) la modélisation de la simulation de la mobilité. Dans [16], les auteurs étudient une méthode de réglage adaptatif pour calibrer les paramètres liés à un algorithme de consensus basé sur Apache Kafka pour des applications de cas d'utilisation spécifiques à la blockchain. Plus précisément, leur méthode est basée sur la théorie du contrôle par rétroaction et vise à ajuster les paramètres liés à son algorithme de consensus afin de faire face à l'afflux soudain et abondant de données et de s'adapter rapidement aux charges de travail actuelles du système. Selon les auteurs de [17], le manque de données peut se produire si le taux de production de données de Kafka dépasse son taux de consommation. Une méthode de délestage est introduite pour limiter les données entrantes et maintenir l'efficacité du système lorsque celui-ci est soumis à des contraintes afin de résoudre le problème de l'épuisement des données. Dans [18], les auteurs décrivent une plateforme de simulation qui permet d'évaluer les futurs cas d'utilisation potentiels de la mobilité. Pour répondre à tous ces besoins et coupler divers outils de simulation dans une co-simulation, Apache Kafka est utilisé comme module de communication.

B. INGÉNIERIE DE LA FIABILITÉ

La transmission fiable et efficace des flux de données est un aspect essentiel de la diffusion de données en réseau. À mesure que les flux de données augmentent en taille et en complexité, le besoin d'algorithmes et de systèmes sophistiqués capables de les gérer s'accroît. Les fondements algorithmiques d'Apache Kafka fournissent un ensemble d'outils et de techniques spécialement conçus pour relever ces défis. La conception algorithmique peut être utilisée pour optimiser la fiabilité et la disponibilité des flux de données, en particulier face à des défaillances ou des perturbations potentielles. Nous présentons les différentes approches qui ont été conçues pour atteindre ces objectifs, y compris les techniques de tolérance aux pannes, de partitionnement, de réplication et d'équilibrage de la charge. Le partitionnement et la réplication sont des techniques clés utilisées pour optimiser la fiabilité et la disponibilité des flux de données dans Kafka. Le partitionnement permet à Kafka de diviser un sujet en morceaux de données plus petits et plus faciles à gérer, qui peuvent être répartis entre plusieurs courtiers. Cela permet de s'assurer que chaque partition peut être traitée indépendamment, ce qui améliore les performances globales et l'évolutivité du système. En outre, la réplication est utilisée pour assurer la tolérance aux pannes et la redondance des données en veillant à ce que chaque partition soit répliquée plusieurs fois, les répliques étant réparties entre plusieurs courtiers.

La tolérance aux pannes pour améliorer la fiabilité est un objectif de conception majeur dans un ensemble d'articles de la littérature. Les auteurs de [19] suggèrent que, parallèlement à la réplication, le point de contrôle de la récupération des messages peut constituer une autre approche de la tolérance aux pannes. En définissant des valeurs idéales d'intervalle de point de contrôle qui ont un effet sur la résilience des données du flux de travail Apache Kafka, ils encouragent l'amélioration de la tolérance aux pannes dans la conception. Les auteurs estiment le coût total des frais généraux après avoir défini l'intervalle de point de contrôle optimal, et ils l'ajustent avec précision en ce qui concerne la maximisation du message perdu.

le taux de récupération. Il a été démontré que l'utilisation de l'intervalle idéal entre les points de contrôle réduit considérablement la quantité de données perdues. D'après l'étude comparative, le système modifié améliore la capacité d'Apache Kafka à récupérer les données. Dans [20], les auteurs étudient les défis de la tolérance aux pannes en cas de reprise après sinistre pour Apache Kafka et introduisent une approche de clusters Kafka redondants et coopératifs dans l'espace pour renforcer la résilience. Dans [21], les auteurs s'attaquent au problème du délai de livraison des messages d'Apache Kafka dans des contextes où des pannes de réseau peuvent survenir, et ils concluent que la taille du lot a un impact direct sur le taux de perte de données, en particulier lorsque la connexion réseau n'est pas stable. Enfin, dans [22], les auteurs étudient l'impact sur les performances de fiabilité de différents paramètres de configuration d'Apache Kafka, y compris la réplication des partitions pour la tolérance aux pannes.

L'analyse comparative de l'architecture peut également aider à mesurer diverses propriétés d'un cluster Apache Kafka. Selon [23], les projections de l'effet sur les performances de diverses configurations d'Apache Kafka ne sont pas toujours exactes. Les auteurs mettent l'accent sur les comportements inattendus constatés dans les opérations de données Kafka. Plus précisément, (i) deux producteurs de données exécutés indépendamment ne doublent pas le taux d'entrée des données par rapport à un seul producteur, comme prévu, et (ii) l'utilisation de la mémoire observée n'a jamais été proche de ses limites pour aucun des scénarios présentés. Cependant, la recherche montre que tous les cas observés ne confirment pas l'hypothèse. Dans [24], les auteurs démontrent que la modification des configurations dans des circonstances normales (telles que l'utilisation de la bande passante du réseau et le taux de service moyen des producteurs Kafka) peut avoir un impact sur les propriétés de livraison des données. Ils utilisent la prédiction de la fiabilité de Kafka en fonction de diverses configurations et conditions de réseau, et définissent deux mesures de fiabilité à prédire, la probabilité de perte de données et la probabilité de duplication de données. Les réseaux neuronaux artificiels sont appliqués au modèle de prédiction et certains paramètres clés sont sélectionnés, ainsi que des métriques de réseau en tant que caractéristiques. Enfin, dans [25], les auteurs présentent la conception d'un cadre de test pour évaluer les aspects de fiabilité d'Apache Kafka afin d'étudier diverses approches de livraison de données dans une qualité de réseau sous-optimale. Deux mesures, le taux de perte de données et le taux de duplication, sont utilisées dans les expériences afin d'évaluer la fiabilité de la livraison de données dans Kafka. Les résultats expérimentaux montrent qu'en cas de retard important du réseau, la taille des données a de l'importance.

V. OPTIMISATION DE L'INFRASTRUCTURE EN RÉSEAU

Alors que de plus en plus d'applications se déplacent vers les environnements de cloud computing et de edge computing, le besoin d'infrastructures de réseau efficaces et évolutives devient de plus en plus critique. Apache Kafka offre une plateforme robuste pour le streaming de données en réseau, permettant l'échange et le traitement de données en continu à travers des systèmes distribués. Cette section se concentre sur les capacités d'optimisation de l'infrastructure en réseau d'Apache Kafka, en mettant l'accent sur sa conception de réseau et son architecture Edge- to-Fog-to-Cloud (E2F2C). Tout d'abord, nous explorons comment Apache Kafka peut être utilisé pour optimiser le réseau.

Nous examinons les diverses considérations de conception de réseau qui doivent être prises en compte lors de la mise en œuvre d'Apache Kafka, telles que le pipelining, l'altération en aval/en amont, ainsi que la mise en file d'attente et le filtrage intelligents. Ensuite, nous nous concentrons sur la façon dont Kafka peut être utilisé pour optimiser le streaming et le traitement des données à travers les systèmes distribués dans le continuum E2F2C. Nous examinons comment les différentes options architecturales peuvent être utilisées pour améliorer l'efficacité du traitement des données et réduire la latence, en poussant le traitement des données à la périphérie et en tirant parti des ressources fog et cloud. En outre, nous examinons les différentes technologies et techniques utilisées pour mettre en œuvre l'architecture E2F2C, telles que la consommation de ressources par des tiers, l'allocation de ressources en direct, ainsi que le déchiquetage de la charge.

A. CONCEPTION DU RÉSEAU

La conception du réseau est une tâche cruciale pour assurer le fonctionnement efficace des systèmes en réseau. Dans le contexte du streaming de données en réseau avec Apache Kafka, la conception du réseau fait référence a u processus de conception de la topologie et des protocoles du réseau afin de garantir une transmission fiable des données et une faible latence. Elle implique des décisions telles que le nombre et l'emplacement des courtiers, la configuration des producteurs et des consommateurs Kafka et le choix des protocoles de communication. Une conception efficace du réseau est essentielle pour obtenir un débit élevé et une faible latence dans les applications de diffusion en continu. Dans cette section, nous passerons en revue les principaux travaux sur la conception de réseaux pour Apache Kafka et discuterons des différentes approches et techniques proposées dans la littérature.

Dans [26], les auteurs séparent le processus de diffusion en continu en deux parties différentes et évaluent les délais pour deux déploiements différents afin de déterminer si une application de diffusion en continu typique est suffisamment intensive en termes de réseau pour bénéficier d'u n e interconnexion plus rapide. En outre, ils cherchent à savoir si le volume du flux de données d'entrée a un effet sur les caractéristiques de latence du pipeline de diffusion en continu et, dans l'affirmative, quelle est la comparaison entre les différentes étapes du pipeline de diffusion en continu et les différentes interconnexions de réseau. Dans [27], les auteurs découvrent que, plutôt que d'être poussé et répliqué dans des emplacements en aval, le filtrage des grands ensembles de données est mieux réalisé dans un emplacement partagé en amont. Ils font évoluer Apache Kafka pour mener des opérations sur des données restreintes, en reprenant certains processus opérationnels des applications en aval, afin d'illustrer les avantages d'une telle stratégie. Par rapport à Kafka standard, leur méthode obtient de meilleurs résultats que quatre conceptions populaires de pipeline analytique avec une surcharge minimale. Dans [28], les auteurs présentent un mécanisme de streaming pour les réseaux optiques basé sur l'architecture et les protocoles Kafka, afin de distribuer efficacement les mises à jour d'état et de réseau conformément à l'accord de mise en œuvre du streaming de l'Open Networking Foundation Transport API à venir. Le mécanisme proposé est validé et évalué expérimentalement. Dans [29], l e s auteurs conçoivent un système de messages distribués d'Apache Kafka pour prendre en charge les messages distribués à grande échelle entre les contrôleurs SDN. Le système proposé mesure le temps de traitement des messages de Kafka et de la file d'attente de messages existante et évalue ses performances. Afin d'obtenir des résultats efficaces, les auteurs ont conçu un système de messagerie distribuée Apache Kafka.

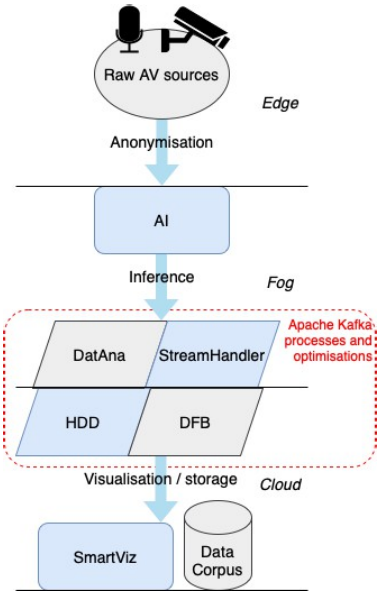

FIGURE 5 : Exemple [31] d'implémentation d'Apache Kafka dans le continuum E2F2C.

Pour corriger le taux d'encombrement, les auteurs de [30] conçoivent un modèle de consommateur qui améliore Apache Kafka grâce à des méthodes de filtrage et de mise en file d'attente fiables, évolutives et intelligentes. Même en cas de congestion extrême, le modèle de consommateur introduit est capable de garantir qu'aucune donnée n'est perdue tout en limitant le nombre de retransmissions. Les résultats expérimentaux montrent qu'en termes de fiabilité, de taux de consommation et de rendement, l'approche proposée est plus performante que le consommateur Kafka traditionnel.

B. CONTINUUM E2F2C

Le continuum E2F2C est une architecture informatique distribuée qui intègre différentes couches de traitement et d'analyse des données générées par les appareils périphériques. L'architecture se compose de trois couches : Edge, Fog et Cloud. La couche Edge représente les dispositifs physiquement situés à proximité de la source de données, tandis que la couche Fog est une couche distribuée qui fournit des capacités de traitement intermédiaires entre les couches Edge et Cloud. La couche Cloud est la couche finale qui est responsable du stockage et du traitement des données collectées à grande échelle. Cette architecture fournit un cadre pour le traitement distribué, qui devient de plus en plus important à mesure que les organisations collectent davantage de données et exigent un traitement plus rapide et plus efficace. En tirant parti des avantages de chaque couche, l'architecture peut permettre le traitement des données en temps réel, l'analyse et la prise de décision. Dans cette section, nous passerons en revue les travaux liés au continuum Edge-to-Fog-to-Cloud dans le contexte d'Apache Kafka. Une mise en œuvre indicative d'Apache Kafka dans le cadre du continuum E2F2C est fournie dans [31] et son architecture respective est présentée à la Fig. 5.

Dans [32], les auteurs présentent une méthodologie pour augmenter l'architecture de référence actuelle basée sur Kafka d'un cas d'utilisation E2F2C en la généralisant et en permettant d'étendre les ressources d'un cluster Apache Kafka donné avec des ressources supplémentaires situées sur des propriétaires de nuages industriels tiers, en tirant parti des fonctionnalités des produits sur étagère.

au-delà des déploiements en périphérie. Un cadre appelé Kafka On Hadoop (KOHA), créé et mis en œuvre par les auteurs de [33], offre aux utilisateurs une approche rapide, facile et efficace pour créer une application distribuée à grande échelle basée sur Kafka qui s'exécute au-dessus d'un cluster Hadoop. L'architecture alloue des ressources pour lancer des producteurs et des consommateurs et construit et exécute automatiquement des courtiers Kafka à la demande. Les utilisateurs n'ont pas besoin de comprendre le modèle de programmation YARN ou d'essayer de mettre en place un cluster Kafka afin d'utiliser le cadre pour adopter Apache Kafka. La capacité d'Apache Kafka à tolérer les défaillances du réseau est examinée dans [34]. Les auteurs notent qu'Apache Kafka fait preuve d'une certaine tolérance aux fautes pour les problèmes de réseau dans diverses configurations, et ils notent également certains de ses inconvénients. En outre, ils établissent une référence pour la tolérance aux défaillances du réseau qui peut être utilisée pour comparer d'autres systèmes de diffusion en continu distribués. Dans [35], les auteurs évaluent diverses configurations et mesures de performance d'Apache Kafka afin d'aider les utilisateurs à éviter les goulets d'étranglement et à tirer parti des meilleures pratiques pour un traitement efficace des flux. Dans [36], les auteurs présentent un moteur de délestage basé sur Apache Kafka qui fonctionne lorsque le délai dépasse le point de référence et gère rapidement le débordement en rejetant les données dans le producteur Apache Kafka. La répartition de la charge s'est avérée efficace à la fois pour les sources uniques et les sources multiples dans les tests utilisant Apache Storm.

VI. TRAITEMENT DES DONNÉES

Avec la prolifération des données dans les systèmes modernes, la nécessité de disposer de systèmes de gestion des big data efficaces et évolutifs est devenue de plus en plus importante. Kafka, avec sa capacité à gérer de grands volumes de données en temps réel, offre une solution potentielle à ce problème, car il fournit une solution évolutive et tolérante aux pannes pour le traitement des données en temps réel. Dans cette section, nous explorons les différentes façons dont Kafka peut être utilisé pour la manipulation et le traitement des données. Nous commençons par examiner comment Kafka peut être intégré aux cadres d'apprentissage machine (ML) pour permettre des calculs efficaces et évolutifs basés sur le ML. Plus précisément, nous nous penchons sur Kafka- ML, une bibliothèque open-source qui facilite l'intégration de Kafka avec des frameworks ML populaires tels que TensorFlow, Keras et Scikit-learn. Nous discutons également des avantages de l'utilisation de Kafka-ML pour le traitement de données à grande échelle et présentons quelques cas d'utilisation réels. Dans la deuxième sous-section de cette section, nous nous concentrons sur la gestion des big data à l'aide d'Apache Kafka. Nous explorons les différentes caractéristiques de Kafka qui le rendent adapté à la gestion des big data, notamment son architecture distribuée, sa tolérance aux pannes et son évolutivité. Nous abordons également certains des défis associés à l'utilisation de Kafka pour la gestion des big data, tels que la sérialisation des données et l'intégration avec d'autres technologies big data.

A. ML ET KAFKA-ML

Au cours des dernières années, la ML s'est imposée comme un outil puissant pour l'analyse et le traitement de grandes quantités de données. Cependant, l'approche traditionnelle du traitement par lots n'est pas toujours adaptée aux applications en temps réel qui nécessitent des réponses immédiates aux données entrantes. Pour y remédier, un nouveau paradigme de traitement en temps réel a été mis au point.

La ML est apparue, qui implique le traitement de flux de données à l'aide d'algorithmes de ML. Apache Kafka est devenu un choix populaire pour le traitement des flux de données en temps réel et l'intégration avec les cadres de ML. Kafka-ML est une puissante bibliothèque open-source qui permet l'intégration d'algorithmes de ML dans les flux Kafka. Cette intégration permet de développer des applications de ML en temps réel, qui peuvent fournir des réponses immédiates aux données entrantes.

Dans [37], Kafka-ML est présenté comme un cadre open source de pointe qui permet la gestion de pipelines de ML par le biais de flux de données. Les utilisateurs peuvent rapidement construire des modèles de ML, les entraîner, les tester et les déployer pour des inférences à l'aide de l'interface utilisateur Web accessible et conviviale de Kafka-ML. Grâce à l'utilisation de technologies de conteneurisation, Kafka-ML et les composants qu'il utilise sont entièrement gérés, ce qui garantit leur portabilité, leur facilité de distribution et d'autres caractéristiques telles que la tolérance aux pannes et la haute disponibilité. La dernière étape est l'introduction d'une méthode unique de gestion et de réutilisation des flux de données, qui pourrait supprimer la nécessité de systèmes de fichiers ou de stockage de données.

Dans [38], les auteurs étendent le cadre Kafka-ML pour prendre en charge l'administration et la mise en œuvre de réseaux neuronaux profonds distribués dans le continuum E2F2C, car ils affirment que Kafka-ML ne prend pas en compte la distribution des modèles de réseaux neuronaux profonds. Afin de fournir des prévisions rapides, ils ont également réfléchi à la possibilité d'inclure les sorties anticipées dans les couches de l'E2F2C. En modifiant et en déployant leur modèle dans trois scénarios distincts, ils évaluent son potentiel. Par rapport à une mise en œuvre uniquement dans le nuage, les expériences montrent que Kafka-ML peut considérablement augmenter le temps de réaction et le débit en distribuant les modèles DNN sur le continuum nuage-objets.

Les auteurs de [39] fournissent un cadre technologique qui combine les avantages de BranchyNet (une architecture de réseau neuronal dans laquelle des branches latérales sont ajoutées à la branche principale, le réseau neuronal de base original, pour permettre à certains échantillons de test de sortir plus tôt) avec l'idée architecturale Edge-Cloud pour permettre des prédictions d'IA tolérantes aux pannes et à faible latence. La mise en œuvre et l'évaluation de cette méthodologie permettent d'évaluer les avantages de l'utilisation d'un réseau neuronal distribué (DDNN) tout au long du continuum allant du Cloud aux objets. Les statistiques recueillies démontrent une amélioration du temps de réponse de 45,34 % par rapport à un déploiement dans le nuage uniquement. En outre, cette proposition offre une extension Kafka-ML qui réduit la rigidité lors de la gestion et du déploiement du DDNN dans le continuum du Cloud aux objets.

Dans [40], les auteurs proposent une méthode d'analyse automatisée de nouvelles hétérogènes par le biais d'algorithmes de traitement d'événements complexes et de ML. Au départ, le contenu des actualités est diffusé en continu à l'aide d'Apache Kafka, stocké dans Apache Druid, puis traité par un mélange de techniques de traitement du langage naturel et de ML non supervisé.

Dans [41], les auteurs proposent de créer KafkaFed, une architecture de communication évolutive basée sur un réseau centré sur l'information, afin de permettre la méthode d'apprentissage fédéré (AF). L'infrastructure basée sur la mise en réseau centrée sur l'information (ICN) permet une recherche rapide des données pour les utilisateurs mobiles.

tout en surmontant les inconvénients des conceptions client-serveur conventionnelles pour FL, qui utilisent un routage basé sur le contenu ou sur le nom. Pour garantir une livraison efficace et fiable des données, celles-ci sont stockées dans des nœuds intermédiaires du réseau ICN. Dans un cadre simulé, une preuve de concept pour l'architecture de communication KafkaFed est créée et testée. Avec seulement 32 clients, le cadre proposé a surpassé l'architecture FL basée sur un serveur client, ou FLOWER, en termes de performances. Il présente également divers avantages supplémentaires en termes d'évolutivité, de stabilité et de sécurité.

Alors que les plateformes de médias sociaux continuent d'être confrontées au problème des faux comptes et des activités automatisées, il existe u n intérêt croissant pour l'utilisation des techniques de ML afin d'identifier et de suivre les robots sociaux en temps réel. Récemment, les auteurs de [42] ont exploré l'utilisation d'Apache Kafka en tant que puissante plateforme de streaming pour traiter les données des médias sociaux et appliquer des algorithmes de ML pour identifier les activités suspectes. Pour identifier en temps réel les robots sociaux sur Twitter à l'aide de la ML, ils utilisent Apache Kafka pour diffuser les données de l'API de Twitter. Ils utilisent les détails des profils comme caractéristiques. Pour prévoir le type de données entrantes, une méthode de ML est utilisée.

B. GESTION DES DONNÉES MASSIVES (BIG DATA)

L'augmentation croissante des données a conduit au développement de diverses techniques et outils pour gérer les données de manière efficace. Apache Kafka, en tant que plateforme de streaming distribuée, fournit une solution robuste pour gérer de grands volumes de données en temps réel de manière tolérante aux pannes et évolutive. Kafka permet l'ingestion, le traitement et la diffusion de données à haut débit et à faible latence, ce qui est essentiel pour la gestion des données volumineuses. En outre, Kafka s'intègre à divers cadres de traitement des données volumineuses tels qu'Apache Spark, Apache Storm et Apache Flink pour prendre en charge le traitement et l'analyse des données à grande échelle.

Dans [43], les auteurs développent Apache Kafka en proposant un système distribué de reconnaissance d'événements complexes conçu au-dessus des flux Apache Kafka. L'objectif principal du système est de raisonner sur la sémantique des opérateurs de flux Kafka. Pour ce faire, les auteurs le conçoivent avec les opérations d'abstraction de la construction, de la transformation et de la composition d'événements.

Les auteurs de [44] présentent KSQL, un moteur SQL en continu pour Apache Kafka. Pour le traitement en flux sur Apache Kafka, KSQL offre une interface SQL simple et entièrement interactive, éliminant la nécessité d'écrire du code dans un langage informatique comme Java ou Python. Open-source, distribué, évolutif, fiable et en temps réel, KSQL est aussi... Les agrégations, les jointures, le fenêtrage, la sessionnalisation et une large gamme d'autres fonctions avancées de traitement de flux sont prises en charge. Grâce aux fonctions définies par l'utilisateur (UDF) et aux fonctions agrégées définies par l'utilisateur (UDAF), il est extensible. Étant donné que KSQL est construit à partir de l'API Kafka Streams, il offre la garantie d'une livraison exacte, une évolutivité linéaire, une tolérance aux pannes et la possibilité de fonctionner en tant que bibliothèque sans cluster séparé.

La référence [45] se concentre sur la grammaire du modèle de connaissance des objets et sur la manière dont elle peut être utilisée pour analyser et reconnaître les métadonnées dans les flux basés sur Kafka. La grammaire proposée est plus flexible et s'avère plus efficace dans les cas suivants

combinée à d'autres techniques NLP connues. Les auteurs affirment que le modèle proposé correspond mieux à la compréhension NLP et projettent les avantages de leur utilisation dans les modèles de gestion des connaissances parce que la grammaire teste la véracité de l'encadrement des phrases (dans différentes langues) et parce que chaque langue a ses propres origines et connotations.

Dans [46], les auteurs décrivent l'architecture d'un moteur RSP basé sur des cadres Big Data de pointe, en particulier Apache Kafka et Apache Spark. Ensemble, ils permettent le développement d'un moteur RSP prêt à la production qui garantit une haute disponibilité, une évolutivité, une tolérance aux pannes, une faible latence et un débit élevé. Ils soulignent également à quel point il est plus facile de développer des applications complexes nécessitant des bibliothèques pour l'apprentissage automatique, le traitement des graphes, le traitement des requêtes et le traitement des flux grâce au cadre Spark.

Dans [47], les auteurs comparent Apache Kafka et RabbitMQ en utilisant les propriétés fondamentales des systèmes pub/sub, et se livrent également à une évaluation qualitative et empirique des qualités partagées par les deux systèmes. Ils soulignent également les caractéristiques uniques que chacun de ces systèmes possède. Ils s'efforcent de guider le lecteur à travers un tableau de détermination pour choisir l'architecture appropriée à ses besoins spécifiques après avoir énuméré une sélection de cas d'utilisation qui conviennent le mieux à RabbitMQ ou à Kafka.

Les trois chemins de données les plus intensifs en termes de réseau (sortie d'enregistrements, duplication d'enregistrements et consommation d'enregistrements) sont accélérés à l'aide d'un accès direct à la mémoire à distance par l'extension Apache Kafka KafkaDirect, présentée par les auteurs de [48]. Ils étudient les options de conception, telles que les procédures d'accès direct à la mémoire à distance à employer pour utiliser pleinement la communication déchargée. Ils utilisent des demandes unilatérales d'accès direct à la mémoire à distance dans l'architecture qu'ils proposent pour réaliser une véritable communication sans copie sans utiliser de tampons intermédiaires dans les serveurs Kafka, ce qui se traduit par un faible retard et une communication à haut débit.

Afin de réduire la latence de la diffusion de données basées sur le contenu, les auteurs de [49] proposent une toute nouvelle forme de sujet dans Kafka, appelée fat topic. L'idée de base qui sous-tend l'amélioration des performances de transfert avec les fat topics est qu'après avoir fait correspondre un événement avec un ensemble d'abonnements, l'événement peut être stocké avec la liste des correspondances dans un topic, au lieu de transférer l'événement d'un topic à de nombreux topics. En outre, ils modifient le code Kafka pour introduire des API de consommateur et de producteur pour l'accès aux fat topics. Pour évaluer l'efficacité des "fat topics", ils ont effectué des tests complets. Les résultats de l'expérience indiquent que, par rapport au sujet Kafka initial, le fat topic peut réduire la latence de la diffusion d'événements basés sur le contenu d'environ
3,7 fois.

Les auteurs de [50] suggèrent d'utiliser l'informatique en flux pour rationaliser le flux de travail de reséquençage du génome, en améliorant son efficacité et sa tolérance aux pannes. Afin de permettre une composition simple et une inclusion dans les pipelines préexistants basés sur YARN, ils divisent les premières étapes du reséquençage génomique en deux parties : l'étape de l'analyse du génome et l'étape de l'analyse de l'ADN.

Roboticised microscopy
Medical data analysis

Aquaculture monitoring
CO2 monitoring

TABLEAU 3. Convergence cyber-physique

Article	How	Why
Internet of Things		
[51]	Robot Operating System	Robotic integration
[52]	Integrated streaming	
[53]		
[54]	Distributed messaging	Image recognition
[55]	Data analysis throughput	Video analysis
[56]	Micro-workflows	Digital twin
Vehicles and mobility		
[57]	MQTT integration	C-ITS messages
[58]	Video streams	Vehicle tracking
[59]	Architectural design	Autonomous vehicles
[60]		Mobility simulation
Environmental use cases		
[61]	Heterogeneous streaming	Environmental data analytics
[62]	Homogeneous streaming	
[63]		
[64]	LSTM neurons	Weather forecast
[65]	Stream storage	Meteo sensor data storage
[66]	Distributed cluster processing	Seismic waveform data

en deux modules discrets et spécialisés (prétraitement et alignement). Nous composons ensuite ces modules de manière souple par le biais de la communication sur les flux Kafka. La solution proposée est ensuite vérifiée empiriquement à l'aide de données réelles, et il est démontré qu'elle s'adapte de manière approximativement linéaire.

VII. CONVERGENCE CYBER-PHYSIQUE

La convergence des systèmes physiques avec les applications pilotées par les données (tableau 3) est une tendance émergente qui gagne rapidement du terrain dans divers domaines tels que les soins de santé, l'énergie, les transports et la surveillance de l'environnement. Cette section du document examine le rôle d'Apache Kafka dans l'intégration des systèmes physiques avec les applications basées sur les données, en mettant particulièrement l'accent sur la convergence cyber-physique. Nous examinons comment Kafka peut être utilisé pour permettre la communication et la coordination en temps réel entre les systèmes physiques et les applications pilotées par les données, facilitant ainsi le développement d e systèmes intelligents capables d'apprendre des événements du monde réel et d'y répondre de manière opportune et efficace. Cette section est divisée en trois sous-sections, chacune d'entre elles se concentrant sur un domaine d'application spécifique. La première sous-section explore la manière dont Kafka peut être utilisé pour gérer et traiter les données générées par des capteurs et d'autres dispositifs IoT, tels que des robots ou des dispositifs dotés d'une fonction image/vidéo/VR. La deuxième sous-section examine comment Kafka peut être utilisé pour permettre la communication et la coordination en temps réel entre les véhicules et d'autres éléments de l'infrastructure de transport. La troisième sous-section examine comment Kafka peut être utilisé pour gérer et traiter les données générées et collectées pour les applications environnementales, dans lesquelles Apache Kafka est une option courante pour le transfert de données, et examine comment Kafka peut être utilisé pour permettre la surveillance et l'analyse en temps réel des données environnementales, facilitant ainsi le développement de systèmes intelligents pour la surveillance et la gestion de l'environnement. Dans l'ensemble, cette section met en évidence le rôle important qu'Apache Kafka peut jouer pour favoriser la convergence cyber-physique dans toute une série de domaines d'application.