Évaluation des performances des réseaux d'oracles Chainlink dans les blockchains

par

Emna HAMMEMI

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE DES TECHNOLOGIES DE L'INFORMATION
M. Sc. A.

MONTRÉAL, LE 03 DÉCEMBRE 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Abdelouahed Gherbi, directeur de mémoire
Département de génie logiciel logiciel et des TI, École de technologie supérieure

M. Kaiwen Zhang, codirecteur
Département de génie logiciel logiciel et des TI, École de technologie supérieure

M. Mohamed Faten Zhani, président du jury
Département de génie logiciel logiciel et des TI, École de technologie supérieure

M. Chamseddine Talhi, examinateur externe
Département de génie logiciel logiciel et des TI, École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 26 NOVEMBRE 2021

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

J’honore toutes les bonnes causes qui m’ont soutenu au cours de ce programme de recherche.

Je tiens à remercier M. Abdelouahed Gherbi et à exprimer ma gratitude pour ses efforts, son encadrement et pour ses contributions à la réussite de ce travail.

Je remercie M. Kaiwen Zhang pour sa disponibilité, son encouragement et ses précieux conseils tout au long de mon projet de recherche qui ont contribué à éclairer ma réflexion.

J’adresse un grand merci à mes parents, à ma famille et M. Faouzi Bahloul, maître de conférences à l’Écode Nationale d’Ingénieurs de Tunis, pour leur soutien moral, suivi apprécié et encouragement durant ma maîtrise.

Évaluation des performances des réseaux d'oracles Chainlink dans les blockchains

Emna HAMMEMI

RÉSUMÉ

L'apparition des contrats intelligents dans le secteur Blockchain a attiré une variété de domaines. Ceci a aidé Blockchain à étendre de manière flexible son territoire au-delà des marchés financiers pour couvrir l'Internet des objets, la chaîne d'approvisionnement, etc. C'est pourquoi les contrats intelligents deviennent plus exigeants en termes de données du monde réel. Par conséquent, les oracles ont été proposés pour assurer la transmission des données entre Blockchain et hors chaîne. Chainlink, en particulier, est une solution décentralisée d'oracles, émergente, qui propose de transférer des données de sources externes directement vers des contrats intelligents. Dans ce mémoire, nous proposons d'explorer théoriquement et expérimentalement Chainlink. Les principaux composants de Chainlink sont le nœud et les adaptateurs Chainlink, nous étudions alors les meilleures options pour les opérateurs de nœuds et les utilisateurs de Chainlink à adopter dans leurs systèmes. En particulier, nous démontrons comment les nœuds Chainlink se comportent lors de la variation d'un ensemble de ses paramètres internes et externes. Cela inclut les caractéristiques matérielles ainsi que les éléments logiciels. Enfin, nous avons mis en place un modèle Chainlink décentralisé basé sur les services Cloud et évalué les performances globales du système en termes de temps de réponse et de coût total des services Cloud, Ethereum et Chainlink.

Mots-clés: Blockchain, Oracles, Contrats intelligents, Chainlink.

Performance Evaluation of Oracle Networks in Blockchains

Emna HAMMEMI

ABSTRACT

The emergence of Smart Contracts in the Blockchain sector attracted a variety of domains. This helped Blockchain flexibly extend its territory beyond financial markets to cover the Internet of Things, supply chain, etc. For this reason, Smart Contracts become more demanding in terms of real-world data. Consequently, oracles were proposed to ensure data transmission between Blockchain and hors chaîne. Chainlink, in particular, is an emerging decentralized oracle solution that offers to transfer data from external data sources right to Smart Contracts. In this thesis, we propose to theoretically and experimentally explore Chainlink. The main components of Chainlink are the Chainlink node and adapters, we therefore investigate the best options for node operators and Chainlink users to adopt in their systems. In particular, we demonstrate how Chainlink nodes behave when varying a set of its internal and external parameters. This includes hardware as well as software elements. Finally, we implemented a decentralized network of Chainlink oracles based on Cloud services and evaluated the overall performance in terms of response time and total cost of Cloud, Ethereum and Chainlink services.

Keywords: Blockchain, Oracles, Smart Contract, Chainlink.

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

P2P	Peer To Peer
EVM	Ethereum Virtual Machine
API	Application Programming Interface
TEE	Trusted Execution Environments
IT	Information Technology
IAAS	Infrastructure as a Service
PAAS	Platform as a Service
SAAS	Software as a Service
AWS	Amazon Web Services
FAAS	Function as a Service
SGX	Software Guard Extensions
SC	Smart Contract
EA	External Adapter
GUI	Graphical User Interface
CDF	Cumulative distribution function
HDD	Hard Disk Drive
SSD	Solid State Drive

INTRODUCTION

La blockchain est sans aucun doute une technologie émergente qui remet en question les méthodes traditionnelles de stockage et de transmission des données. Il s'agit d'un registre public numérique basé sur un vaste réseau distribué qui garantit la confiance, l'intégrité des données, l'authenticité, la transparence et la haute disponibilité. La blockchain est principalement une chaîne de blocs qui stocke des informations numériques. Ils sont liés entre eux par des hachages cryptographiques pour former une chaîne liée. La blockchain a été introduite avec Bitcoin (Nakamoto, 2008). Il était d'abord dédié à la sécurisation des transactions financières basées sur la cryptomonnaie. Quelques années plus tard, les applications de Blockchain ont été largement étendues pour aller au-delà des services financiers et couvrir les chaînes d'approvisionnement, les jeux vidéo, la santé, l'Internet des objets et plus encore.

L'essor remarquable de la technologie blockchain dans des domaines au-delà des cryptomonnaies peut être attribué au développement de contrats intelligents introduits par Ethereum (Buterin, 2013). Les contrats intelligents sont définis comme des programmes informatiques ou des scripts, exécutant automatiquement un ensemble d'actions lorsqu'un ensemble de conditions d'accord prédéterminées sont satisfaites. Les actions effectuées peuvent être aussi simples que payer un abonnement, retirer des frais ou obtenir des informations, et peuvent être aussi délicates que les systèmes de vote du gouvernement et les soins de santé. Ainsi, les contrats intelligents obligent la Blockchain à être beaucoup plus exigeante en termes d'informations externes, dans certains cas, les données hors chaîne deviennent indispensables. Pour pouvoir concevoir des systèmes industriels opérationnels basés sur la blockchain, les contrats intelligents doivent acquérir des données du monde extérieur, tels que les domaines de l'automobile, du gouvernement, de la vente au détail et de transport. Cependant, compte tenu de la nature de l'environnement isolé de Blockchain, les contrats intelligents ne peuvent pas remplir cette tâche à eux seuls. En fait, la sécurité du système de réseau Blockchain en dehors de ses protocoles cryptographiques et de son architecture distribuée est renforcée par sa ségrégation absolue avec le monde hors chaîne

(données et événements existant en dehors du réseau Blockchain) (Beniiche, 2020). Cependant, cette séparation bloque de manière critique les opérations de ces contrats. Par conséquent, une question concernant le mode d'acquisition des données est soulevée. C'est exactement la raison pour laquelle les oracles ont été conçus.

Les oracles peuvent être représentés comme une entité tierce qui relie le système Blockchain (en chaîne) au monde extérieur (hors chaîne). C'est un pont qui fournit des informations externes aux contrats intelligents. Maintenant que les oracles deviennent un membre essentiel de l'environnement Blockchain, il est important de mentionner l'impact qu'ils apportent sur l'écosystème Blockchain. L'introduction d'un middleware unique entre un système décentralisé et une autre entité ramène l'ensemble du processus de communication à la centralisation (Al-Breiki, Habib Ur Rahman & Svetinovic, 2020).

Malgré ses avantages essentiels pour les contrats intelligents, l'introduction d'un middleware unique entre une Blockchain et une entité externe est critique, car elle permet la circulation d'entrées non fiables et malignes vers la Blockchain. Ce dilemme entre l'importance des oracles et les risques de sécurité potentiels a conduit à la proposition de diverses solutions d'oracles. Chacune de leurs implémentations propose une approche différente pour aborder le problème de confiance des oracles. En particulier, Chainlink est une solution d'oracles émergente et en constante évolution, lancée en mai 2019. Chainlink est basée sur la décentralisation et la réputation afin de maintenir la sécurité lors de la livraison de données aux contrats intelligents (Adler et al., 2018). Un système décentralisé basé sur Chainlink est un réseau de nœuds qui utilise des adaptateurs pour récupérer des données externes. L'analyse des performances d'oracles est fondamentale en vue de décider de la meilleure configuration à opter pour une efficacité optimale.

D'autre part, le Cloud Computing est évidemment une technologie répandue, qui devient indispensable, et qui peut couvrir tous les domaines industriels. Il fournit des services importants aux

entreprises au niveau de l'infrastructure, plateformes et applications, offrant des fonctionnalités considérables telles que la haute disponibilité, la maintenance et les faibles coûts. En particulier, Serverless Computing est une catégorie émergente de Cloud Computing où le côté serveur est complètement invisible pour les utilisateurs et est maintenu par le fournisseur Cloud.

Dans cette thèse, nous proposons les contributions suivantes :

1. Nous proposons une étude complète de l'architecture, des mécanismes et des caractéristiques de Chainlink en détail : son principe, ses composants et ses services.
2. Nous évaluons expérimentalement les performances d'un seul oracle Chainlink en ce qui concerne ses paramètres matériels et logiciels. Nous étudions les meilleures options pour les opérateurs de nœuds et les utilisateurs de Chainlink à adopter dans leurs systèmes. En particulier, nous démontrons comment les nœuds Chainlink se comportent lors de la variation d'un ensemble de ses paramètres internes et externes décisifs tels que la méthode de déploiement du nœud, les type d'adaptateurs et d'initiateurs, les fournisseurs Ethereum et la configuration matérielle.
3. Nous proposons la conception, mise en œuvre et évaluation d'un réseau décentralisé d'oracles Chainlink. Notre système est basé sur des services de Cloud Computing. Nous avons analysé la performance globale en termes de temps de réponse et de coût total des services Cloud, Ethereum et Chainlink.

À notre connaissance, il s'agit du premier travail consacré à l'étude expérimentale de Chainlink basée sur la mise en œuvre de systèmes réels dédiés, et qui mènent vers une évaluation des composants principaux de Chainlink puis d'une architecture décentralisée basée sur Chainlink.

Cette thèse est structurée de cette manière : dans le premier chapitre, nous étudions correctement les différentes technologies que nous avons exploitées dans ce projet de recherche puis présentons les travaux associés. Le deuxième chapitre est divisé en deux sections principales. Dans la

première, nous étudions en détail l'architecture et les caractéristiques de Chainlink : son principe, ses composants et ses services. Nous avons consacré la deuxième partie à présenter notre étude expérimentale de Chainlink à l'échelle centralisée puis décentralisée. Dans le dernier chapitre, nous détaillons le montage et la méthodologie de nos expériences puis présentons et discutons les résultats de notre évaluation.

CHAPITRE 1

ÉTAT DE L'ART

Ce chapitre sera d'abord consacré à présenter des concepts généraux sur Blockchain, ses principaux mécanismes, plateformes, oracles et contrats intelligents. La deuxième section est dédiée au Cloud computing, en particulier, ses services à bases de serveurs et ses services sans serveurs.

1.1 Définition des concepts de base

1.1.1 Blockchain et les systèmes distribués

La numérisation mondiale des transactions a transformé tous les processus industriels et commerciaux et a conduit à révolutionner les périmètres traditionnels et à la naissance de nouveaux domaines et technologies innovants comme l'Internet des objets, le Cloud Computing, la Business Intelligence, etc. Cette rénovation mondiale a récemment atteint l'industrie financière, par l'introduction de Blockchain, une nouvelle perspective de la sécurité, de la résilience et de l'efficacité des systèmes. «The Financial Times (2016) defines Blockchain as "network of computers, all of which must approve a transaction has taken place before it is recorded, in a 'chain' of computer code. The details of the transfer are recorded on a public ledger that anyone on the network can see.» (Ahram, Sargolzaei, Sargolzaei, Daniels & Amaba, 2017). Il a été introduit pour la première fois par Satoshi Nakamoto, avec Bitcoin (Nakamoto,)dans le cadre des transactions financières. La proposition consistait à abolir la dépendance vis-à-vis des autorités financières centralisées en distribuant les confirmations de transactions sur un réseau distribué. Il s'agit principalement d'une chaîne de blocs où sont stockées les informations numériques, sur la base d'un grand réseau distribué pair à pair (P2P). Un modèle de réseau pair à pair est le concept élémentaire au sein des réseaux distribués, où les nœuds sont capables de communiquer directement sans qu'un tiers soit nécessaire pour maintenir ou administrer la communication.

De cette manière, les transactions sont traitées par des parties non fiables (également appelées mineurs) pour produire un travail de confiance.

1.1.2 Sécurité dans les Blockchains

Les blockchains sont formées par blocs. Toutes les informations gérées par les chaînes de transaction sont stockées dans ces blocs. Selon la plateforme , un bloc peut stocker plusieurs à des milliers de transactions. Chacune d'entre elles possède une empreinte cryptographique, appelée hachage. Lorsqu'un nouveau bloc est créé, il stocke le hachage du bloc précédent. Ainsi, chaque bloc est composé des données et du hachage du bloc précédent. La seule exception concerne le premier bloc créé dans la Blockchain, le bloc genesis, car il n'a pas de bloc précédent, alors pas de hachage précédent. De cette façon, chaque bloc est lié à son précédent, cela crée en effet une chaîne de blocs. Une fois qu'une transaction est sur la Blockchain, elle est immuable et permanente. Ce mécanisme est illustré par la figure 1.1 (Ajao, Agajo, Adedokun & Karngong, 2019). La sécurité de la Blockchain is prinicpalement assurée par la cryptographie, la décentralisation et les mécanismes de consensus.

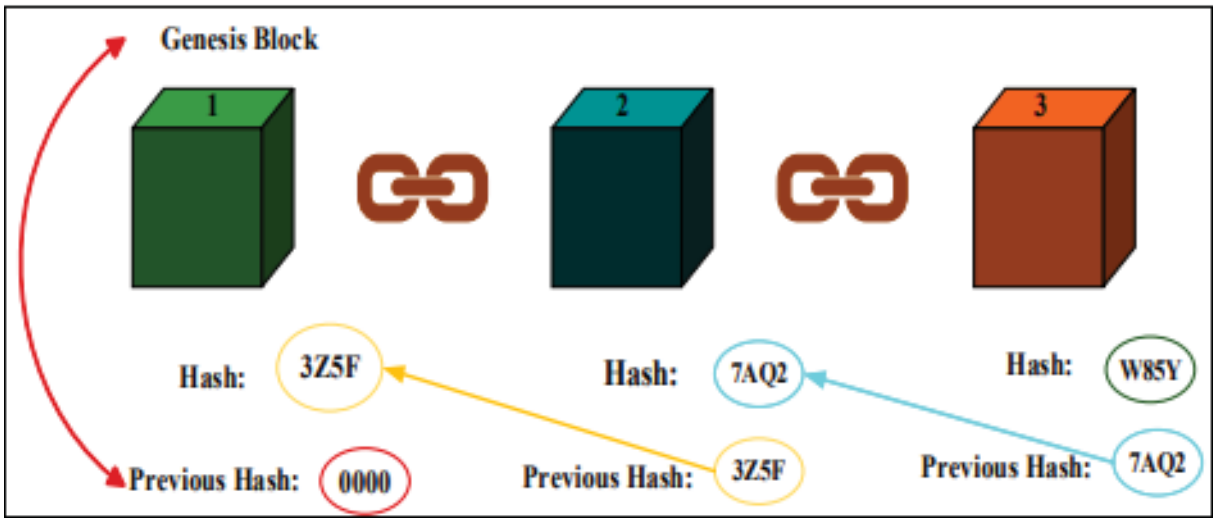

Figure 1.1 Liaison de hachage à l'intérieur du réseau Blockchain
Tirée de Ajao (2019, p. 5)

- La cryptographie : Toutes les transactions blockchain sont sécurisées par cryptographie. Chaque bloc contient essentiellement une clé unique et privée qui peut être vérifiée avec une clé publique. S'il y a un changement dans les données liées à la transaction, la clé unique de bloc devient invalide. En conséquence, le bloc est supprimé de la chaîne.

- La décentralisation : La technologie Blockchain est sécurisée principalement grâce à sa décentralisée et la distribution de son réseau. Il n'y a pas de point de défaillance unique, ce qui rend la corruption beaucoup plus difficile. Le piratage d'une partie du système ne peut pas affecter les autres parties. Cependant, dans le cas d'une blockchain privée, cet avantage est en partie perdu car ils ont un seul point de contrôle et un nombre limité de nœuds. Cela empêche les utilisateurs d'apporter des modifications au grand livre. Les organisations exploitent ce type de blockchain pour leur usage interne, car cela permet à l'entreprise de contrôler ses propres processus.

- Les mécanisme de consensus : Puisque le réseau des Blockchain est basé sur la distribution, toute la technologie blockchain fonctionne selon un modèle de consensus, entre des membres du réseau, afin de vérifier qu'une transaction a eu lieu et la légitime. La plupart des modèles de consensus fonctionnent sur des protocoles qui incluent une preuve de travail, une preuve d'enjeu, une preuve d'autorité, etc. Dans la section suivante, nous décrivant une méthode de consesnsus très connue, utilisée par Bitcoin : la preuve de travail.

1.1.3 Les Plateformes de Blockchain

1.1.3.1 Bitcoin

Bitcoin est la première plateforme, fondée en 2008, qui implémente le concept distribué de Blockchain, dédié à la crypto-monnaie. Ainsi, cela représente un nouveau défi pour les institutions financières traditionnelles entièrement dépendantes d'un tiers de confiance pour valider et traiter les transactions. Au contraire de Bitcoin, qui envoie la validation de la transaction sur un réseau P2P distribué. Pour assurer l'intégrité au sein du réseau d'entités non nécessairement fiables, les

mineurs doivent s'entendre selon un mécanisme de consensus sur la validation de chaque bloc (Nofer, Gomber, Hinz & Schiereck, 2017). Il s'agit d'une plateforme (publique) sans permission, où tout le monde est automatiquement autorisé à rejoindre le réseau en tant que mineur.

- **Le mécanisme de consensus** : Selon Swanson, ce mécanisme de consensus "is the process in which a majority (or in some cases all) of network validators come to agreement on the state of a ledger. It is a set of rules and procedures that allow maintaining coherent set of facts between multiple participating nodes" (Swanson, 2015). Toutes les plateformes Blockchain n'utilisent pas le même algorithme de consensus, pour Bitcoin, il utilise la preuve de travail. En fait, chaque mineur (un nœud Blockchain désireux de valider des blocs dans le réseau), reçoit chronologiquement un ensemble de transactions, puis les collecte dans un bloc. Chaque nœud résout ensuite l'algorithme de consensus pour le nouveau bloc et se déplace pour le diffuser au sein du réseau. Le réseau est informé du premier bloc qui a été miné et procède à la vérification de sa validation.

- **Preuve de travail :** Étant donné que Bitcoin fonctionne sur un réseau distribué et sans autorisation, auquel tout le monde peut se joindre, des problèmes de consensus ainsi que de sécurité sont soulevés concernant la falsification des données et les mineurs malveillants. La preuve de travail vient pour résoudre cette situation critique. En effet, pour valider des blocs, ce mécanisme oblige les mineurs à se concurrencer pour résoudre un problème cryptographique complexe, coûteux en ressources, chronophage, de durée 10 min pour Bitcoin, mais facile à vérifier une fois résolu. Ce mécanisme de consensus assure la sécurité au sein du réseau. Si un nœud malveillant veut traiter un bloc avec des données falsifiées et l'ajouter à la Blockchain, il doit dépasser 51% du réseau afin de valider le bloc modifié, ce qui est actuellement peu probable car il est extrêmement coûteux en termes de ressources.

1.1.3.2 Hyperledger Fabric

Contrairement à Bitcoin, la plateforme Blockchain sans autorisation, Hyperledger Fabric [1] est une plateforme avec autorisation, où tous les types de nœuds doivent être identifiés par un fournisseur de services d'adhésion (MSP) et seuls les nœuds authentifiés peuvent traiter les transactions et traiter des blocs (Xu et al., 2021). Il s'agit d'un système distribué et open source initialement fondé par Linux Foundation et actuellement maintenu par IBM et Linux Foundation (Nasir, Qass, Abu Talib & Bou Nassif, 2018). Contrairement à Bitcoin, Fabric ne fournit pas de crypto-monnaie. Il convient à la gestion, au stockage et au partage de données dans un contexte d'entreprise où un groupe de nœuds autorisés contrôle l'ensemble du système distribué. Fabric est un sous-projet du projet parent Hyperledger.

1.1.3.3 Ethereum et les contrats intelligents

Semblable à Bitcoin, Ethereum est une plateforme publique Blockchain, ayant son propre jeton de crypto-monnaie, nommé Ether. elle a été lancé en 2015. Il permet aux utilisateurs d'exécuter leurs programmes sur l'environnement Ethereum. Les contrats intelligents sont un élément clé élémentaire de l'environnement Ethereum. En fait, Ethereum a relancé le concept de contrats intelligents qui avait été proposé pour la première fois en 1994 par Nick Szabo (Wang et al., 2018). Ethereum n'est pas la seule implémentation basée sur des contrats intelligents, Hyperledger Fabric et Corda peuvent également être utilisés dans ce contexte. En particulier, les contrats intelligents Ethereum sont écrits en langage Solidity et sont exécutés dans Ethereum Virtual Machine (EVM). Solidity est un langage de haut niveau qui fournit des sous-contrats et des bibliothèques. Les bibliothèques sont censées offrir des outils qui facilitent le développement ou l'optimisation des contrats, tandis que les sous-contrats aident les développeurs à incorporer des connexions orientées objet comme l'héritage entre les contrats intelligents (Oliva, Hassan & Jiang, 2020). La figure 1.2, présente le mécanisme d'exécution du code d'un contrat intelligent écris en Solidity et le cycle est le suivant : Le code solidity est compilé via solc, le compilateur

[1] Hyperledger Fabric : https ://www.hyperledger.org/use/fabric

solidity. Il génère le Bytecode, formé d'opcodes, puis est exécuté par l'EVM (Bouicho, Mezroui & Oualkadi, 2020).

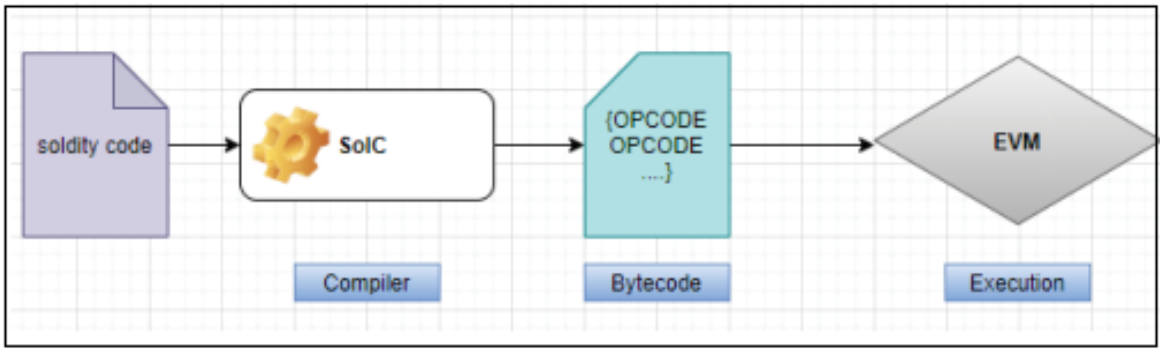

Figure 1.2 Le processus d'exécution du code Solidity dans Ethereum Blockchain
Tirée de Bouicho (2020, p. 3)

- Impact des contrats intelligents sur les Blockchains : Contrairement à Bitcoin, Ethereum ne cible pas spécifiquement le domaine financier, il est plutôt conçu pour aller au-delà de la crypto-monnaie et couvrir autres domaines d'application, ce qui étend fortement ses cas d'utilisation pour s'adapter à l'Internet des objets, la santé, le commerce, etc. et la Blockchain devient de plus en plus populaires à mesure que des secteurs comme le gouvernement, la santé et le secteur immobilier découvrent les avantages. Une partie de cette expansion remarquable est due grâce aux contrats intelligents. Ces derniers sont basés sur le concept If/Then. Totalement génétique et standard, ils peuvent être appliqués sur une grande variété d'applications et cas d'utilisation. Ils sont constitués d'un ensemble de fonctions qui peuvent être appelées séparément, pour exécuter un ensemble d'instructions. Ils sont définis comme des programmes auto-exécutables, automatiques, basés sur un ensemble d'accords entre les entités concernées. Entièrement personnalisés, les contrats intelligents permettent aux parties intéressées d'établir leurs propres règles de manière flexible. Dans la figure suivante, nous présentons un cas d'utilisation des contrats intelligents dans le domaine des applications quotidiennes, qui reflète l'émergence des contrats intelligents aussi bien que l'importance des données du monde réel pour les Blockchains. Dans ce cas, elle consiste à utiliser la Blockchain pour le paiement automatique, en quelques clics, basé sur une simple reconnaissance de visage, de l'achat d'une application mobile à partir de Google Play ou App Store par exemple.

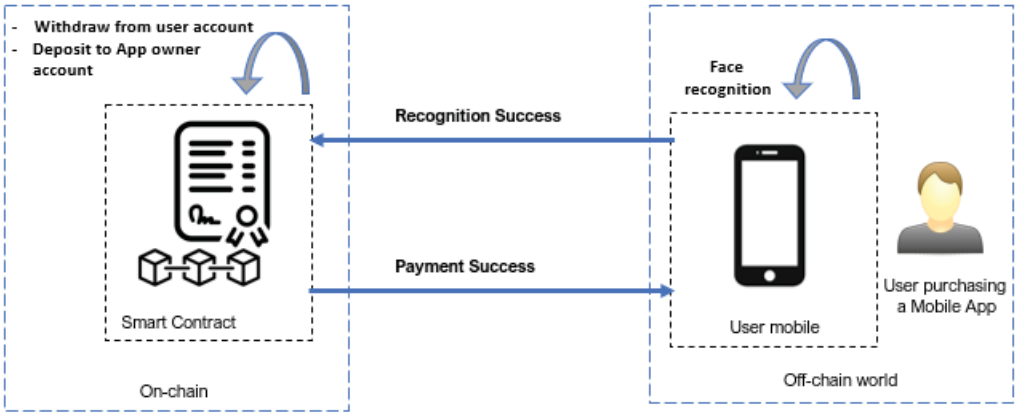

Figure 1.3 Cas d'utilisation des contrats intelligents : paiement d'une application mobile

Dans ce cas d'utilisation, si l'utilisateur est authentifié avec succès à travers une reconnaissance de visage, c'est-à -dire, il est autorisé à utiliser le compte Google associé, alors une notification d'identification avec succès sera envoyée à la Blockchain avec d'autres informations en liaison avec le compte monnaie de l'utilisateur, à l'intention d'un contrat intelligent spécifique. À la réception de ces données du monde réel, autrement dit, une fois que les conditions du contrat sont satisfaites, il va automatiquement lancer l'exécution des actions correspondantes. Dans ce cas, les actions consistent à effectuer le paiement par crypto-monnaie, en enlevant ce montant du compte monnaie de l'utilisateur et le verser dans le compte monnaie des propriétaires de l'application achetée. Ainsi, à travers ce cas d'utilisation, l'utilisateur paie effectivement et automatiquement à travers la Blockchain en utilisant la crypto-monnaie, sans avoir à utiliser les moyens manuels, tels que les cartes crédits.

- Interactions avec les contrats intelligents : Une fois déployé sur la Blockchain, un utilisateur Ethereum peut interagir avec les contrats intelligents. Leur déploiement s'effectue via une transaction appelée transaction de création de contrat. Dans ce cas, le contrat obtient son adresse, un identifiant unique de 20 octets dans le réseau Ethereum. L'interaction avec les contrats intelligents se produit via ses appels de fonctions invoqués par l'utilisateur à travers une transaction. Chaque transaction (appel de fonction) consomme une quantité d'unités de gaz qui dépend du nombre d'instructions à exécuter. Le gaz est l'unité utilisée pour calculer le coût de

chaque transaction Blockchain. Par conséquent, afin d'envoyer la transaction, l'utilisateur doit spécifier le prix du gaz, exprimé en Eher, qui présente le montant de la devise que l'utilisateur est prêt à offrir. Aussi, la limite de gaz, qui présente le maximum d'argent que l'utilisateur veut payer. Ainsi, l'utilisateur paiera le prix du gaz multiplié par la consommation de gaz . Certes, plus les prix du gaz proposés par l'utilisateur sont élevés, plus les mineurs sont motivés à traiter la transaction rapidement. De toute évidence, chaque fonction du contrat intelligent produit des frais de transaction différents qui dépendent du prix du gaz et du nombre ainsi que de la complexité des instructions à exécuter au cours de l'exécution (Oliva et al., 2020).

- **Vérification des contrats intelligents :** Afin d'améliorer la transparence et la confiance, Ethereum fournit aux développeurs de contrats un moyen de publier le code source, qui sera disponible pour toute la communauté Ethereum. En effet, lorsqu'un contrat intelligent est déployé, seule sa version bytecode sera conservée sur la Blockchain. Le développeur dans ce cas peut décider de publier ou non le code source. Etherescan, la principale plateforme d'exploration de blocs et d'analyse pour Ethereum, offre un moyen de télécharger le code et de vérifier son authenticité. Une fois que le développeur a téléchargé le code, Etherscan compile le code puis vérifie si le bytecode généré est identique à celui stocké sur la Blockchain. Une fois vérifié, le code source sera publié (Olivia et al., 2020).

1.1.4 Les oracles de Blockchain

1.1.4.1 Définition et objectifs des oracles

La confiance est l'élément clé dans les transactions financières. Chaque système de stockage met en place ses propres moyens de sécurité, de confidentialité et de confiance. La blockchain, basée sur la décentralisation, assure la sécurité grâce à son solide consensus et ses algorithmes cryptographiques vulgarisés sur l'ensemble de son réseau. Les oracles, des composants nouvellement impliqués dans l'écosystème Blockchain, sont obligés de maintenir de manière robuste le niveau de sécurité élevé offert par le système distribué Blockchain (Kochovski and al., 2019). Un oracle est une passerelle qui connecte les contrats intelligents au monde extérieur. Il ne fait pas partie

des contrats ni des informations externes elles-mêmes. Il s'agit d'un composant complètement différent qui transmet les données entre le demandeur et la source de données (Weisheng and al., 2021). Les données requises peuvent être de plusieurs types : informations de transaction, données en temps réel, données de capteurs, etc. De plus, de nombreuses sources de données sont prises en charge : sites Web, bases de données, toute API, etc.

Compte tenu du besoin important des oracles pour alimenter les contrats intelligents avec des données externes, il existe une tendance dynamique qui a conduit à l'élaboration de plusieurs solutions d'oracle récemment. C'est pourquoi plusieurs approches ont été proposées. Selon Beniiche, les oracles peuvent être catégorisés en fonction de plusieurs critères : le type de source de données (logiciel, matériel ou humain), le sens de circulation de l'information (entrant ou sortant) et enfin en fonction de son architecture (centralisée ou décentralisée). Dans ce travail, nous nous concentrerons sur deux catégories principales : l'architecture centralisée/décentralisée et les oracles à flux entrant/sortant (Beniiche, 2020).

1.1.4.2 Les types d'oracles

- Oracles centralisés/Décentralisés : Élargissant potentiellement la portée des applications des contrats intelligents, les utilisateurs de Blockchain, en particulier les entreprises, sont préoccupés par l'architecture Oracle. En d'autres termes, le principe d'architecture des oracles est une caractéristique clé qui joue un rôle important dans la définition du niveau de sécurité du système.

Un oracle centralisé est un système à un composant chargé de relier les contrats intelligents au monde extérieur (Caldarelli, 2020). Toutes les tâches Oracle sont contrôlées, gérées et exécutées par cette entité unique. Chaque solution propose un moyen de sécurité différent. En particulier, Provable [2] est basé sur un ensemble de preuves d'authenticité et Town Crier fonctionne sur des environnements d'exécution sécurisés (TEE). Cette centralisation, malgré ses tentatives pour instaurer un bon niveau de confiance contre les attaques, pose tout de même immédiatement des problèmes de sécurité. En fait, s'appuyer sur un seul composant pour incorporer des données

[2] Documentation Provable : https ://docs.provable.xyz/

dans le réseau Blockchain peut exposer le système Blockchain à des pannes et des défaillances. Cela réduira sensiblement la tolérance aux pannes et la haute disponibilité du système. De plus, un seul contrôle malin sur cette seule entité est suffisant pour exposer l'ensemble du système basé sur Blockchain à de vraies menaces. en outre, l'inclusion d'un seul système de sécurité basé sur la confiance (un oracle centralisé) dans l'écosystème Blockchain est en contradiction avec l'absence de confiance, c'est le concept fondamental de Blockchain. Il est également inévitable de mentionner que les oracles centralisés tentent principalement d'assurer l'intégrité des données dans ses 3 états différents : stockées (à l'intérieur de l'oracle), en cours de calcul (à l'intérieur de l'oracle) et en transmission (au sein du réseau : de l'oracle au contrat intelligent). Cependant, ce mécanisme suppose que les données reçues de la source de données sont entièrement fiables et valables à utiliser. Ainsi, les oracles centralisés ne permettent pas de vérifier et d'assurer la confiance des informations de la source de données elle-même. Il s'agit néanmoins de l'une des fonctionnalités d'oracles décentralisées, pour assurer la sécurité de la transmission des données aux contrats intelligents ainsi que la vérification de l'exactitude des informations de la source de données.

En fait, le concept de décentralisation au sein des oracles est similaire à la base de Blockchain. Un grand réseau décentralisé d'oracles est un groupe d'oracles qui sont chacun chargés d'obtenir les informations requises et de les transmettre au contrat intelligent qui a lancé la demande. Ainsi, la théorie de base derrière ce concept est en effet de se concentrer sur la fourniture d'informations via de nombreuses entités non fiables plutôt que de se concentrer sur une seule entité de confiance. Cette pratique renforce l'intégrité et l'exactitude des informations. Il existe de nombreuses plateformes oracles décentralisées, telles que Chainlink et Witnet (De Pedro, Levi & Cuende, 2017) qui sont également des solutions basées sur la réputation. Il est important de mentionner que pour Chainlink en particulier, outre la décentralisation des oracles, un seul oracle a la capacité de demander des informations à différentes sources de données. En d'autres termes, Chainlink définit deux niveaux de décentralisation : la décentralisation d'oracles ainsi que la décentralisation des sources de données. C'est l'une des raisons pour lesquelles Chainlink est désormais une solution émergente d'oracles dédiée à la transmission sécurisée d'informations

du monde réel aux contrats intelligents basés sur une décentralisation à deux niveaux. En outre, un autre mécanisme technique utilisé par les solutions d'oracles décentralisées pour améliorer la sécurité, c'est la réputation.

Le mécanisme de réputation dans les oracles basés sur la réputation fonctionne de manière à ce que les oracles soient choisis pour répondre aux requêtes des clients en fonction d'un ensemble de métriques fiables qui illustrent leurs performances. Certains fournisseurs d'oracle donnent la possibilité aux clients de choisir l'oracle avec lequel ils souhaitent travailler tandis que d'autres fournisseurs d'oracle distribuent eux-mêmes les requêtes entre les oracles. Dans les deux sens, la sélection d'oracle dépend fortement de sa réputation. Un tel système incite les opérateurs oracle à fournir le meilleur service aux utilisateurs. Selon le fournisseur d'oracle, de nombreuses métriques sont prises en compte : taux de réponse le plus élevé, nombre total de transactions, temps de réponse moyens, etc.

- Oracles à flux entrant/sortant : Lors du choix du fournisseur d'oracle avec lequel travailler, il est essentiel de définir la direction des informations : est-ce de la chaîne vers l'extérieur ou de l'extérieur vers la chaîne ou les deux. Jusqu'à présent dans le mémoire, nous avons mis l'accent sur une seule direction : du monde extérieur vers les contrats de la Blockchain. L'oracle remplissant cette tâche est un oracle entrant, qui envoie des données à l'intérieur de la Blockchain. Un exemple de ceci est un oracle qui fournit des informations de suivi de vente au détail en temps réel à un contrat intelligent.

Sinon, les oracles sortants sont destinés à transmettre des données des contrats intelligents vers le monde extérieur. Un exemple d'oracle sortant est celui qui envoie l'état d'un dépôt de fonds réussi à un site Web d'apprentissage en ligne à des fins d'abonnement.

La figure 1.4 illustre un aperçu du travail des oracles au sein de l'écosystème Blockchain. Il présente les 2 architectures d'oracles existantes : centralisée et décentralisée.

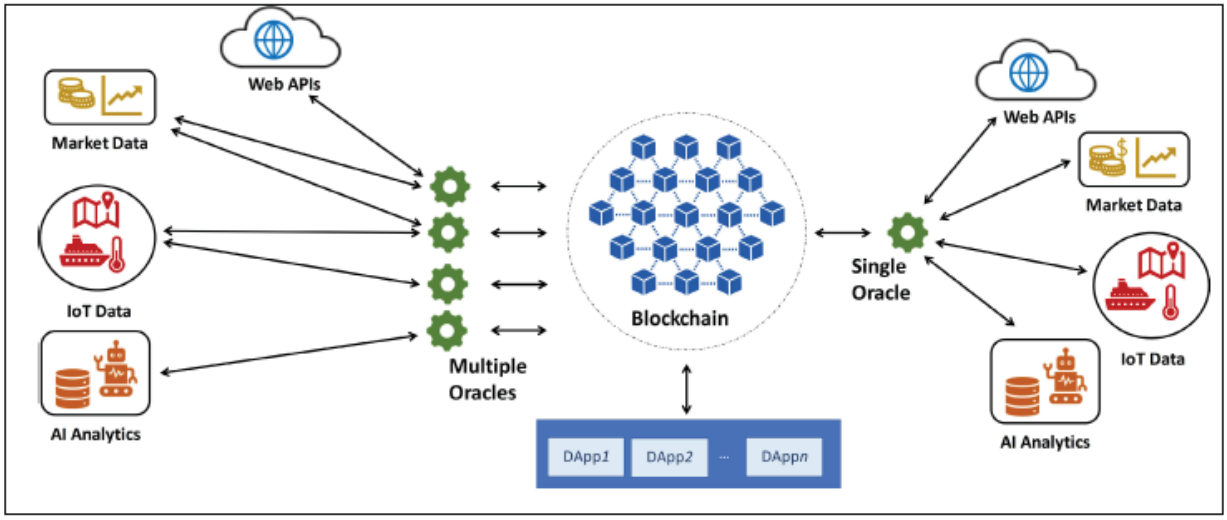

Figure 1.4 Vue d'ensemble du travail d'oracle dans l'écosystème Blockchain

Les deux directions élargissent la portée des contrats intelligents. Ils doublent tous les deux, les domaines que les contrats peuvent intégrer et augmentent leur adaptabilité à l'évolution du marché.

1.1.5 Cloud Computing

1.1.5.1 Cloud computing à base de serveurs

Le Cloud computing est une technologie qui transforme les services et les opérations informatiques tels que les serveurs, le réseau, le stockage et les logiciels en services fournis aux utilisateurs à la demande, par les fournisseurs de Cloud. De nos jours, il existe de nombreux fournisseurs de Cloud connus qui dominent le marché, notamment Microsoft Azure, Google Cloud et Amazon Web Services. La diversité des services à offrir a conduit à les classer en trois grandes catégories :

- **Infrastructure As A Service :** Dans ce modèle Cloud, les fournisseurs offrent aux utilisateurs des ressources d'infrastructure virtuelle telles que des serveurs, des bases de données et réseau, afin que les utilisateurs n'aient pas à acheter des ressources coûteuses, ou à consacrer du temps et

efforts à l'installation, gestion et maintenance. Amazon EC2 et Google Compute Engine sont des exemples de cette classe. IaaS est basé sur le concept de virtualisation des ressources qui permet aux clients de déployer et d'exécuter leurs propres systèmes d'exploitation hôtes et applications personnalisées. La virtualisation dans IaaS est une étape clé vers le déploiement, l'installation et la maintenance distribués, automatiques et évolutifs des logiciels (Prodan & Ostermann, 2009).

- Platform As A Service : Le NIST définit PAAS comme « The capability provided to the consumer is to deploy onto the Cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying Cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment » (NIST, 2011).

Cette catégorie de services fournit aux utilisateurs un ensemble complet d'environnements de développement et de déploiement avec les ressources nécessaires adéquates pour créer des applications sophistiquées. Ce service accompagne toutes les applications Web en offrant tous les outils nécessaires au développement, à la gestion, à la mise à niveau, etc. Google App Engine et Hadoop sont des exemples de cette classe de service.

- Software As A Service : Dans ce modèle Cloud, les fournisseurs Cloud proposent des applications Web entièrement développées, déployées et opérationnelles sur des serveurs Cloud. Ces solutions sont prêtes à l'emploi et les clients n'ont qu'à les acheter puis à accéder à Internet. Gmail et Microsoft Office 365 sont des exemples de SaaS. La couche SaaS est le service le plus visible du Cloud computing, du fait que les applications logicielles sont accessibles directement par les utilisateurs finaux. Ces applications sont déployées et exécutées dans des systèmes Cloud et sont accessibles à partir de divers appareils clients via des interfaces clients simples telles qu'un navigateur Web (Giessmann & Stanoevska-Slabev, 2012).

La figure 1.4 illustre les principales caractéristiques et différences entre les trois catégories.

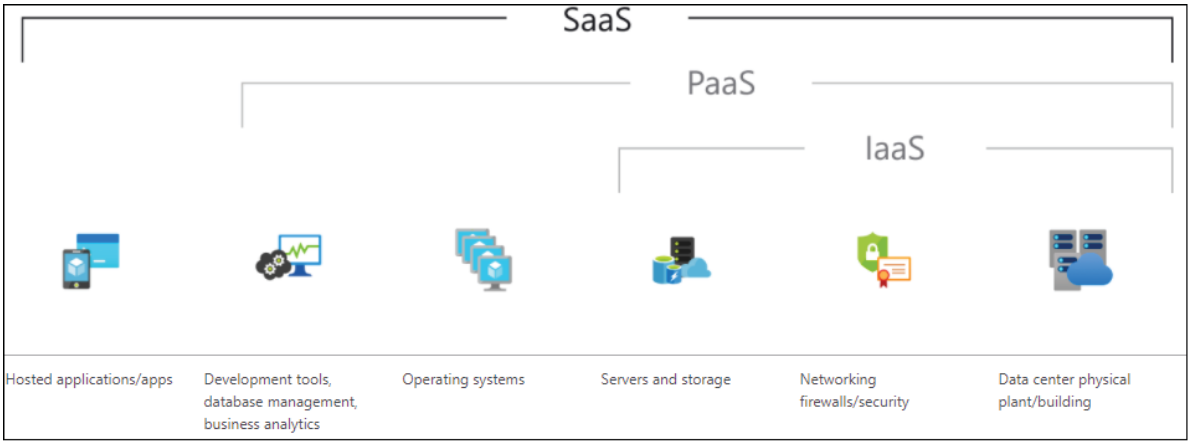

Figure 1.5 Catégories de Cloud Computing

1.1.5.2 Cloud Computing sans serveur

L'infonuagique sans serveur a été proposée pour la première fois par AWS, avec le lancement du service AWS Lambda en 2014. La caractéristique sans serveur ne signifie pas qu'il n'y a pas de serveurs impliqués dans les services. Les serveurs existent toujours, cependant, ils sont entièrement à la charge du fournisseur Cloud. Dans les services ordinaires comme IaaS et PaaS, les clients sont obligés d'allouer en permanence des ressources pour implémenter et faire fonctionner des serveurs même s'ils ne reçoivent qu'une requête par mois, ce qui est évidemment défavorable aux clients. Pour résoudre ce problème, Serverless Computing offre la possibilité de créer des applications basées sur le concept de paiement à l'utilisation, où les clients ne sont facturés que lorsque leurs programmes sont en cours d'exécution à la réception de déclencheurs. En effet, le code des applications s'exécute toujours sur les serveurs, pourtant, le côté serveur est totalement invisible pour le client, la sélection des instances, la mise à l'échelle, le déploiement, la tolérance aux pannes, la surveillance, la journalisation,les mesures de sécurité, etc. sont des tâches à la charge du fournisseur Cloud (Jonas et al., 2019). Dans un tel modèle, les développeurs n'ont qu'à se concentrer sur leurs applications. Il existe de nombreux services basés sur ce concept qui ciblent différentes couches d'une application basée sur le Cloud, comme les serveurs et le stockage. Par exemple, Google Functions et AWS Lambda fournissent des fonctions en tant que service.

- Function As A Service (FAAS) : Ce service sans serveur permet aux clients de diviser leurs applications en fonctions déclenchées par des événements, en d'autres termes, les fonctions ne s'exécuteront que lorsqu'un événement ou une demande l'invoquera. Il permet aux développeurs avec peu à aucune expérience de la logique opérationnelle pour créer, surveiller et invoquer des fonctions Cloud (van Eyk, Iosup & Thömmes, 2017). Selon le fournisseur de Cloud, FaaS propose divers langages de programmation comme Python, NodeJS, Go, Java, etc. Ce service offre zéro administration. Le développeur n'est responsable que de télécharger le code des fonctions. Lorsqu'une fonction est invoquée, le fournisseur Cloud alloue automatiquement les ressources nécessaires pour exécuter le programme et répondre à la requête. Ces ressources seront automatiquement libérées lorsque le travail requis sera effectué. C'est pourquoi il est basé sur le concept de paiement au fur et à mesure, le client ne paie pas lorsque la fonction est inactive. Ce service a notamment incité les clients à utiliser d'autres services Cloud car il est conçu pour être facilement déclenché par les services et l'écosystème de Cloud. De nombreux avantages sont offerts dans le paradigme sans serveur. Par exemple, il fournit une mise à l'échelle automatique, une maintenance automatisée, une haute disponibilité, une efficacité d'utilisation des ressources et une efficacité de charge (par millisecondes). Sur le marché du Cloud, il existe de nombreux frameworks implémentant Serverless Computing tels qu'AWS Lambda, Azure Functions, Google Functions et IBM Cloud Functions. IBM Market Development and Insights a mené une recherche sur l'expérience sans serveur sur le marché. La figure 1.5 présente des statistiques sur les avantages identifiés par les utilisateurs. En effet, la flexibilité de mise à l'échelle, les performances, la réduction des coûts et une latence plus rapide sont des avantages pertinents affirmés par les utilisateurs (IBM Market Development and Insights, 2021).

Dans cette section, nous avons minutieusement étudié les diverses technologies et plateformes que nous avons utilisées dans notre projet de recherche. Notre véritable objectif est autour du réseau d'oracles Chainlink. Comme les oracles appartenaient récemment à l'écosystème Blockchain, il était important de présenter Blockchain, ses plateformes et le concept des contrats intelligents. Nous avons également présenté un aperçu des oracles, de leur objectif et de leurs

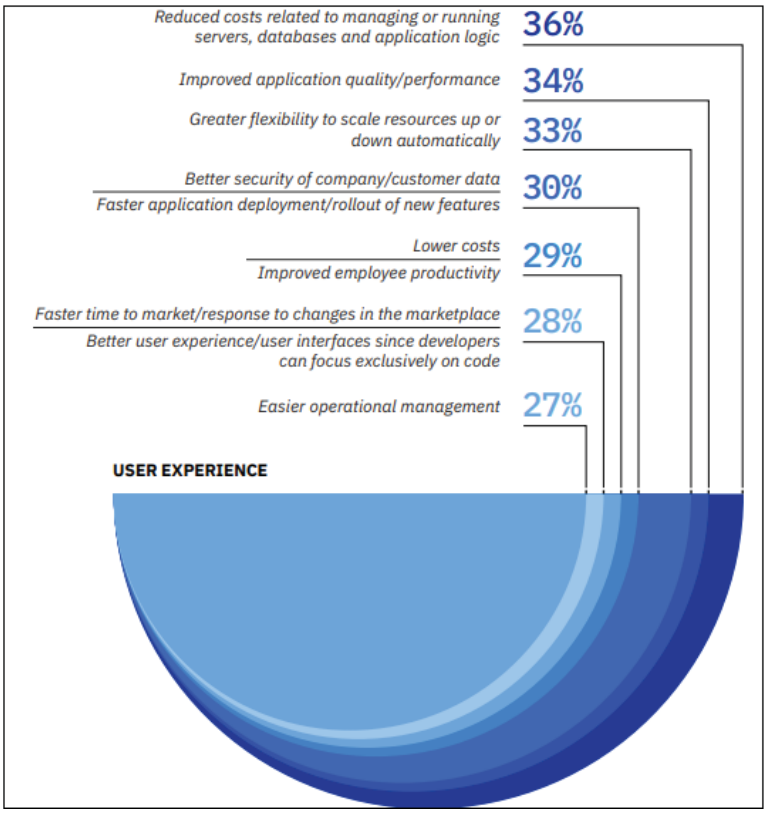

Figure 1.6 Enquête IBM sur l'utilisation sans serveur
Tirée de IBM Market Development and Insights (2021, p. 7)

types. Enfin, comme nous nous appuyions également sur une multitude de services Cloud pour faire fonctionner nos systèmes, nous avons énuméré les principales catégories et services Cloud.

1.2 Revue de littérature

Nous dédions cette section à l'étude d'un groupe de solutions d'oracles qui visent à transmettre en sécurité des données entre le système isolé Blockchain et le monde extérieur. Cette étude est importante pour analyser les architectures et les fonctionnalités proposées. Dans la deuxième partie, nous présenterons les principaux travaux de recherche menés sur l'analyse et l'évaluation des oracles et de Chainlink.

1.2.1 Les plateformes d'oracles

1.2.1.1 Provable

Opérant depuis 2015, Provable appartient à la famille des oracles centralisés. Il peut fonctionner avec une multitude de plateformes Blockchain comme Ethereum, Hyperledger Fabric et Rootstock [3]. Il s'appuie principalement sur des preuves d'authenticité pour garantir l'intégrité des données. En effet, la solution développée par Provable assure la non falsification et l'authenticité des données récupérées de la source de données externe. Ceci est accompli en générant un document appelé preuve d'authenticité et en le renvoyant avec la réponse au contrat intelligent. Différentes technologies peuvent gérer les preuves d'authenticité, notamment Environnements d'exécution de confiance (TEE). Provable est basé sur le concept "If This Then That". En d'autres termes, si un ensemble de conditions est rempli, Provable effectuera un ensemble d'actions. Cela aide la solution à aller au-delà de la Blockchain, pour couvrir également d'autres contextes généraux. Pour valider les requêtes à Provable, le demandeur doit inclure le type de source de données, la requête et le type de preuves d'authenticité.

1.2.1.2 Town Crier

Appartenant aux oracles centralisés, Town Crier (TC) établit la fiabilité au sein des oracles grâce à l'utilisation du jeu d'instructions Intel SGX des Environnements d'exécution de confiance, qui est une nouvelle capacité dans certains processeurs Intel. Les contrats intelligents demandent des données externes au moyen de Town Crier à partir de sources de données externes, plus précisément, de services Internet compatibles HTTPS. En supposant qu'un client fasse confiance à SGX, TC garantit que les données transmises d'un site Web à un contrat intelligent ne sont absolument pas altérées. En d'autres termes, il suffit de faire confiance au SGX d'Intel ainsi qu'à la source de données, pour pouvoir faire confiance à Town Crier et son travail. Il n'est pas nécessaire de faire confiance aux opérateurs de TC ou d'une autre partie (Zhang, Cecchetti & Croman, 2016). La figure 1.6 illustre l'architecture de base de Town Crier.

[3] Rootstock : https ://www.rsk.co/

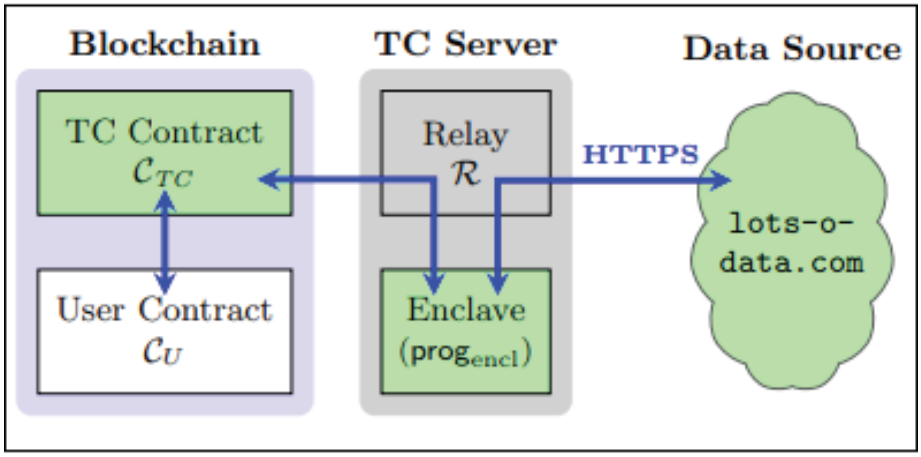

Figure 1.7 L'architecture basique de Town Crier
Tirée de Zhang (2016, p. 4)

Le TC Contract est le composant en chaîne, chargé d'accepter la requête des utilisateurs puis de leur renvoyer la réponse. Le composant Enclave exécute le logiciel TC dans un environnement SGX. Enfin, le Relay gère tout le trafic réseau pour le bien d'Enclave.

1.2.1.3 Witnet

Contrairement aux solution Oracle décrites précédemment, le protocole Witnet exploite la décentralisation afin de connecter les Contrats intelligents à des sources de données externes. Le framework présente un réseau distribué de nœuds, appelés témoins. Il a créé son propre jeton, Wit, avec lequel Witnet récompense ses oracles pour la récupération de données externes en chaîne vers les contrats des utilisateurs. Afin de transmettre en toute sécurité des données dans la Blockchain, un bon nombre de témoins, sélectionnés au hasard, doivent être impliqués afin de récupérer les données d'une ou plusieurs sources Web. Cela peut conduire à former une réponse fiable si la majorité des nœuds ont des données rapportées de manière fiable, grâce à un mécanisme de consensus qui peut détecter l'incohérence. Pour inciter les nœuds à transmettre correctement et fidèlement les données dans la Blockchain, Witnet attribue en fait des tâches aux témoins proportionnellement à leur comportement dans le réseau (De Pedroand Ari, Levi & Cuende, 2017).

1.2.1.4 Chainlink

Étant une solution décentralisée d'oracles, Chainlink est une proposition publique open source pour répondre à un grand besoin de Contrats intelligents, à obtenir des informations externes. Détenant un réseau d'entités indépendantes, le framework Chainlink assure la récupération, l'agrégation et l'envoi des données au contrat demandeur. Chainlink a été proposée en septembre 2017 et officiellement lancé sur Ethereum Mainnet en mai 2019. L'équipe a conçu Chainlink de manière à prendre en charge différentes plateformes Blockchain. Pour l'instant, il est construit et compatible seulement avec Ethereum. Chainlink a son propre jeton, LINK, pour permettre aux oracles d'être payés pour la transmission de données externes aux contrats intelligents. Les clients ou utilisateurs Chainlink font simplement appel à des oracles pour transmettre des données externes en chaîne. Les oracles ou opérateurs de nœuds Chainlink, quant à eux, ils offrent d'exécuter des nœuds Chainlink et d'assurer le transfert de données entre les contrats intelligents et le monde hors chaîne. Reliant les environnements en chaîne et hors chaîne, Chainlink a été conçu avec un haut niveau de modularité. Chaque élément du système est évolutif, de sorte que différents composants peuvent être remplacés à mesure que de meilleures techniques et des implémentations concurrentes apparaissent. Il est également intéressant de noter que l'équipe Chainlink, dans sa stratégie à long terme, vise à intégrer les environnements d'exécution de confiance à ses mesures de sécurité et ainsi à combiner des techniques centralisées à sa solution décentralisée (Ellis, Juels & Nazarov, 2017). Un exemple de réseau d'oracles Chainlink entièrement décentralisé est Chainlink Data Feeds, un service fourni par Chainlink qui permet aux contrats de récupérer facilement le dernier prix d'un actif, une seule demande déclenchera le service. Il travaille actuellement sur plusieurs Blockchains telles que Ethereum, Binance Smart Chain [4], xDai [5] et Solana [6]. Chaque flux de données est géré par un réseau d'oracles Chainlink. Les oracles publient périodiquement les prix, qui sont agrégées par un contrat dédié. Le nombre d'oracles participant au réseau varie d'un Data Feeds à l'autre. Une mise à jour ne se produira que si un nombre minimum d'oracles publient leurs données dans le contrat agrégateur

[4] Binance Smart Chain : https ://www.binance.org/en/smartChain

[5] xDai : https ://www.xdaichain.com/

[6] Solana : https ://solana.com/

correspondant, sinon la mise à jour n'aura pas lieu. Par exemple, la figure 1.7 présente le réseau d'oracles Data Feeds pour ETH/USD [7]. Une fois que 21 oracles sur 31 auront envoyé leurs données, les réponses seront agrégées et le nouveau prix sera publié. Une mise à jour sécurisée est effectuée en chaîne lorsque les données hors chaîne sont modifiées au-delà du seuil d'écart (0,5% pour cet exemple de data Feeds) ou que 3600 secondes se sont écoulées depuis la dernière mise à jour écrite en chaîne.

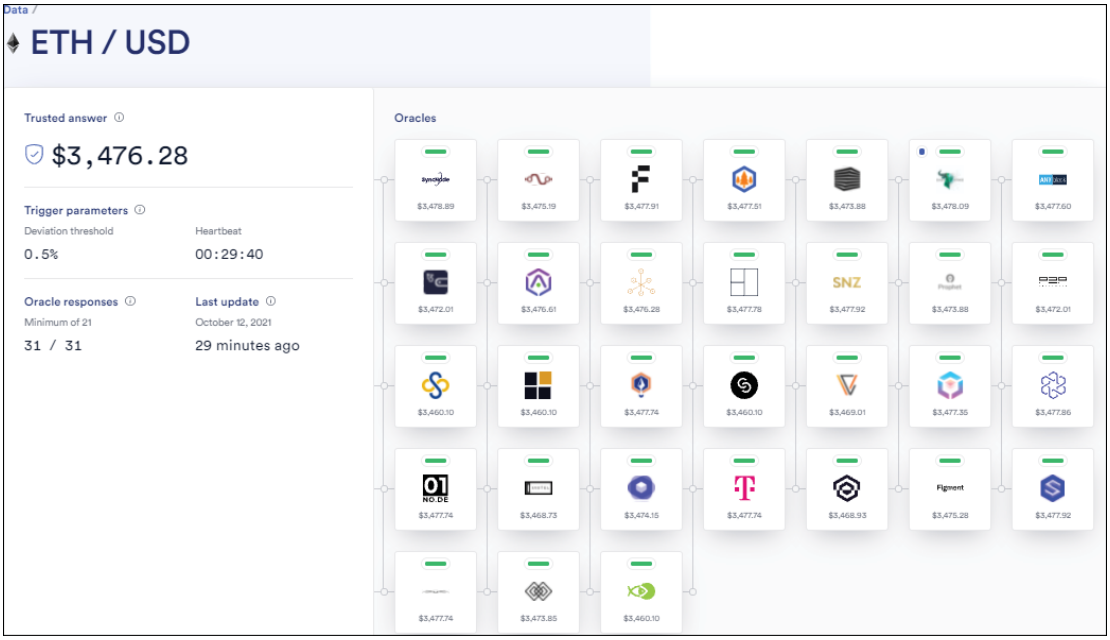

Figure 1.8 Chainlink Data Feeds pour ETH/USD

1.2.2 Travaux antérieurs sur l'évaluation de Chainlink

Étant donné que Chainlink est une solution d'oracles relativement nouvelle et qu'elle évolue constamment, les recherches sur son architecture, ses composants, ses cas d'utilisation et ses performances d'exécution ne donnent toujours pas un panorama complet, aux clients comme aux futurs opérateurs de nœuds, à ce sujet. En fait, à notre connaissance, il existe un seul travail de recherche entièrement dédié à l'analyse des performances de Chainlink, à savoir, Demystifying Pythia : A Survey of ChainLink. D'autres travaux de recherche visant principalement la

[7] Chainlink Data Feeds capturés le 11 octobre 2021 : https ://data.chain.link/

comparaison de différentes solutions d'oracles, nous présentons deux travaux de recherche pertinents dans ce contexte.

1.2.2.1 Demystifying Pythia : A Survey of Chainlink oracles Usage on Ethereum

En fait, Mudabbir et Shi, fournissent des analyses sur l'utilisation des oracles Chainlink et des statistiques du trafic réel du service Data Feeds et l'utilisation de l'API Chainlink. Leur objectif était la collecte du trafic réel de Chainlink pour une période de 18 mois, sur une base de données SQL et l'analyse statistique de son utilisation sur Ethereum Mainnet (Mudabbir & Shi, 2021).

- Travail sur Chainlink Data Feeds : Chainlink Data Feeds (anciennement nommé Price Feeds) est un service qui permet aux utilisateurs de récupérer les derniers prix des actifs (BTC, ETH, LINK, etc.). Pour cette raison, Chainlink était basé sur un réseau décentralisé d'oracles qui demande, chacun, des données de différentes sources de données. La décentralisation fait généralement appel à l'agrégation. Dans cette fonctionnalité, l'agrégation est réalisée via un contrat d'agrégation dédié pour chaque actif. A cet effet, Mudabbir et al. ont exécuté d'abord un client Ethereum pour capturer le trafic données de Data Feeds en modifiant le code Golang pour enregistrer les appels de fonctions internes qui ciblent ces agrégateurs. Au total, ils ont travaillé sur 169 adresses de contrats d'agrégation. Principalement, ils ont enregistré des données telles que le numéro de bloc, le paramètre d'entrée, la valeur et l'adresse d'appel.

- Travail sur Chainlink External APIs : Proposer de récupérer des données à partir de sources externes est le travail le plus fondamental de Chainlink. Il offre alors de récupérer des données à partir des d'API comme des sources Web. Pour pouvoir capturer le trafic Chainlink, les auteurs ont déployé un nœud Ethereum complet et des API Web3 pour détecter les événements Chainlink, notamment les événements ChainlinkFulfilled émis par les contrats clients qui ont initié la demande d'information externe. Les données ciblées décrivent les circonstances de la transaction, telles que : le hachage de la transaction de la requête et de la réponse, le montant de LINK payé, l'adresse du contrat du client et la réponse. Les données collectées ont été stockées sur une base de données MySQL à des fins d'analyse.

Une fois la collecte, le stockage et l'analyse des données terminés, les auteurs ont mentionné que 99,75% des requêtes ont été initiées par les agrégateurs Chainlink. Cela signifie que pendant une période de 18 mois (à partir de mai 2019, date du lancement de Chainlink, jusqu'en octobre 2020), seuls 6 634 requêtes ont été effectués par des clients, et le nombre d'utilisateurs distincts pour les flux de données et les API externes est inférieur à 300. Ils ont également noté que le nombre d'oracles n'augmentait pas de manière remarquable. Ils ont conclu qu'au moment de l'étude, l'écosystème Chainlink sur Ethereum était radicalement vivant grâce au service Data Feeds.

1.2.2.2 Évaluation et comparaison d'oracles

Dans le paragraphe précédent, nous avons détaillé le seul travail entièrement dédié à l'analyse des performances de Chainlink. Cependant, Chainlink était également un thème intéressant pour les chercheurs qui se concentraient sur la comparaison architecturale de plusieurs propositions d'oracles.

- Trustworthy Blockchain oracles : Review, Comparison, and Open Research Challenges :

Dans ce travail de recherche, hamda et al. ont étudié les principales approches et plates-formes d'oracles Blockchain, puis ont discuté des défis ouverts aux oracles concernant la fiabilité. L'étude et la comparaison étaient purement qualitatives. Les premiers aspects de l'analyse portent sur le déploiement des solutions, en chaîne ou hors chaîne. En fait, Provable et Town Crier proposent des systèmes hors chaîne qui transmettent des données externes en chaîne à des contrats intelligents. De l'autre côté, Witnet et ASTRAEA (Adler et al., 2018) proposent des solutions en chaîne. Chainlink, en particulier, effectue une fusion entre des modules en chaîne et hors chaîne pour transmettre des données en toute sécurité. Le deuxième aspect de l'analyse traite le modèle de confiance, où certains oracles sont basés sur une architecture centralisée tandis que d'autres dépendent d'un réseau décentralisé d'oracles. Une autre caractéristique remarquable qui distingue les oracles est les jetons natifs. Par exemple, Witnet et Chainlink

imposent leurs propres jetons, Wit et LINK respectivement, contrairement aux autres oracles.

- **A Study of Blockchain Oracles :** Dans ce travail de recherche, Beniiche étudie théoriquement les oracles Blockchain. Il les a classé en 7 groupes (Beniiche, 2020). Cette classification prend en considération 3 qualités principales : la source des données (matériel, logiciel ou humain), la direction de l'information (entrante ou sortante) et le modèle de confiance (centralisé ou décentralisé). Les oracles logiciels communiquent avec des sources en ligne telles que des sites Web, des serveurs, etc. Les oracles matériels se concentrent sur l'obtention de données du monde physique, telles qu'à partir des capteurs et des objets connectés, puis les transmettent à des contrats intelligents. Les oracles humains s'appuient sur des personnes spécialisées qui vérifient les données puis les envoient en chaîne. Les oracles de calcul sont des oracles qui non seulement relient hors chaîne à en chaîne, mais effectuent également des tâches de calcul supplémentaires et une analyse des données avant d'envoyer les résultats finaux à la Blockchain. Les oracles entrants/sortants sont classés en fonction de la direction des données, si l'oracle transmet des données du monde réel à la Blockchain, il s'agit alors d'un oracle entrant. Le chemin opposé se réfère aux oracles sortants. Les Oracles spécifiques au contrat sont des oracles dédiés à fonctionner avec un seul contrat intelligent. Enfin, les oracles basés sur le consensus récupèrent les données de différentes sources de données, puis exécutent une méthode de consensus telle que la vote pour former la réponse finale à l'intention du contrat intelligent. L'auteur étudie également les oracles centralisés et décentralisés : Provable comme exemple d'oracle centralisé et Chainlink un exemple d'une solution décentralisée, leurs composants et leur architecture globale. En particulier, il explique en détail les mécanismes en chaîne et hors chaîne de Chainlink sur la base du livre blanc de Chainlink (Ellis, 2017).

- **Reliability analysis for Blockchain oracles :** Sin Kuang et al. ont présenté en juillet 2020, une analyse de fiabilité pour un ensemble d'oracles Blockchain actifs où Chainlink faisait partie de la recherche (Kuang Lo, Xu, Staples & Yao, 2020). Le travail a comparé théoriquement les fonctionnalités d'oracle en ce qui concerne le consensus, les caractéristiques de fiabilité, les plates-formes compatibles, les sources de données, l'intervalle de temps et le type d'oracle. Un

extrait de cette comparaison est présenté par le tableau 1. Cette comparaison met l'accent sur les dissemblances entre les oracles et souligne leurs traits et particularités.

Tableau 1.1 Résumé des mécanismes d'oracles de Blockchain

Plateforme	**Oracle**	**Consensus**	**Caractéristiques de fiabilité)**	**Plateformes compatible**	**Source de donnée(s)**	**Intervalle de temps**
Provable	Un seul	N/A	TLS-Notary Proof	Bitcoin, Ethereum, Corda	Une seule	rapide
Town Crier	Un seul	N/A	Intel SGX	Ethereum	Une seule	rapide
Corda	Un seul	N/A	Intel SGX	orda	Une seule	rapide
MS Bletchley	Un seul	N/A	Conteneur sécurisé, Intel SGX	Azure, AWS, Google	multiple	lent
Chainlink	multiple	N-of-M multisignature	aggregation hors chaîne , Reputation	Bitcoin, Ethereum, Hyperledger	multiple	lent
Gnosis	multiple	voting	Ultimate oracle and centralized oracle	Ethereum	multiple	lent

En particulier, Chainlink n'est pas la seule proposition d'oracles à s'appuyer sur la décentralisation. Cependant, parmi les oracles évalués, c'est le seul framework développé pour être compatible avec diverses plateformes Blockchain, ce qui étend sa commodité à un plus grand nombre d'applications. Sinon, chaque solution présente un moyen de sécurité distinct en fonction de l'architecture centralisée/décentralisée, comme les environnements d'exécution de confiance (comme Intel SGX), TLS-Notary Proof, l'agrégation et la réputation. De plus, les auteurs ont mené une analyse quantitative concernant la fiabilité des oracles. Ils ont proposé une approche basée sur l'analyse de l'arbre de défaillance. Selon les résultats, les auteurs ont classé les oracles en deux groupes, les oracles incluant les humains et les oracles excluant les humains. Les résultats les plus élevés, 0,9810 jusqu'à 0,9928 appartiennent à la deuxième catégorie, tandis que les oracles impliquant des humains ont une valeur de fiabilité moins faible, entre 0,88 et 0,92. Certes, l'erreur humaine avait le taux d'erreur le plus élevé.

Les travaux de recherche sur les approches, les mécanismes et les différentes propositions d'oracles n'examinent pas complètement tous les aspects des oracles et les moindres détails. En particulier pour les plates-formes oracles, comme beaucoup d'entre elles sont encore récentes et en constante évolution, il n'existe toujours pas d'étude complète dédiée à l'analyse et à l'évaluation de leurs performances et de leurs efficacités. Dans ce contexte, concernant Chainlink, le seul travail de recherche consacré a visé la collecte du trafic réel et l'analyse statistique de Chainlink. Cela offre à l'utilisateur général un bon aperçu des performances des oracles Chainlink, certes, mais cela n'oriente pas les opérateurs de nœuds concernant les caractéristiques à considérer et les paramètres à sélectionner lors de la mise en œuvre d'un nœud Chainlink et d'un réseau d'oracles. Aussi, il n'explique pas l'impact des éléments internes et externes de l'écosystème Chainlink sur les oracles Chainlink. Par ailleurs, aucune des recherches précédentes n'a conçu et mis en œuvre un système décentralisé basé sur Chainlink et a évalué la performance globale. C'est exactement le travail manquant que nous avons proposé de couvrir.

CHAPITRE 2

CARACTÉRISTIQUES ET ÉTUDE EXPÉRIMENTALE DE CHAINLINK

Dans ce chapitre, nous étudions en profondeur les propriétés de Chainlink. Tout d'abord, nous décrivons ses composants principaux : le nœud Chainlink et les adaptateurs. Nous expliquons également sa fonctionnalité de décentralisation, sur les deux niveaux, puis, nous présentons un aperçu de son mécanisme de réputation. La deuxième partie est consacrée à introduire notre étude pour la partie expérimentale, qui cible l'aspect centralisé aussi bien que l'aspect décentralisé.

2.1 Chainlink : architecture et caractéristiques

Créant un pont entre les systèmes en chaîne et hors chaîne, l'architecture Chainlink est répartie en deux zones : une partie en chaîne et une partie hors chaîne. Le côté en chaîne est chargé d'interagir avec le contrat qui lance la requête (User Smart Contract) et de répondre à la demande, c'est-à-dire détecter la requête, puis soumettre la réponse en chaîne, au contrat demandeur. Alors que le côté hors chaîne est dédié à répondre à la demande ,c'est-à-dire traiter la demande (se connecter à la source externe et récupérer les données concernées). Sur cette base, de nombreux composants fondamentaux des nœuds Chainlink sont impliqués.

2.1.1 Chainlink : architecture

La vue d'ensemble du mécanisme d'appel de Chainlink dans un système Blockchain à travers un contrat intelligent est illustrée par la figure 2.1. L'architecture de Chainlink est composée principalement d'un contrat intelligent, Chainlink Core et les adaptateurs. User Smart Contract est le contrat qui va lancer la requête et external API est la source de données externe à partir de laquelle Chainlink doit récupérer les données demandées.

- Chainlink Smart Contract : Ce composant appartient au système Blockchain. Chaque nœud Chainlink exécute un contrat intelligent (Chainlink Smart Contract) qui doit écouter les journaux

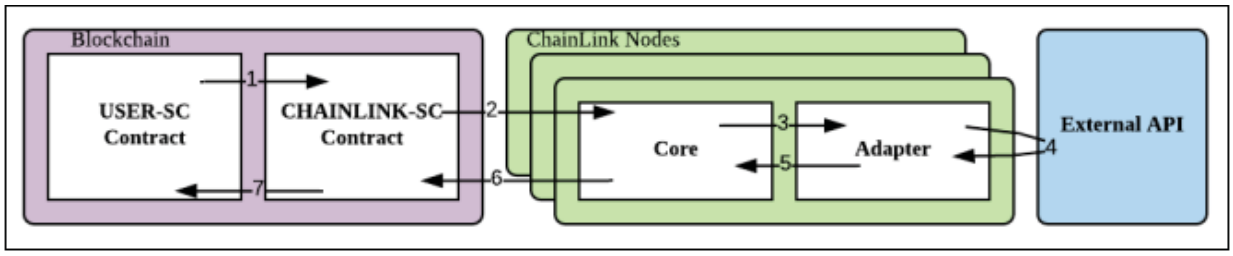

Figure 2.1 Présentation de l'architecture centralisée de Chainlink

de Blockchain afin de détecter les événements émis par les contrats intelligents qui sont à la charge du nœud Chainlink (User Smart Contract). Une fois qu'un événement est détecté, le composant Chainlink Core reçoit les informations appropriées pour effectuer le travail requis.

- Chainlink Core : Chainlink Core (ou logiciel du nœud Chainlink) est responsable de l'interfaçage avec le contrat de Chainlink, de la planification et de l'équilibrage du travail entre ses divers services externes. Pour être connecté à la Blockchain, un nœud Chainlink doit exécuter un client Blockchain. Étant donné que Chainlink est actuellement actif sur Ethereum uniquement, un client Ethereum devient obligatoire. Le travail effectué par les nœuds Chainlink est formaté en tant que tâches. Chaque tâche est un ensemble de spécifications de tâches plus petites, appelées sous-tâches ou aussi adaptateurs de base ou internes (Core adapters) qui sont traitées comme un pipeline. Chaque sous-tâche a une opération spécifique à exécuter, avant de transmettre son résultat à la sous-tâche suivante et d'atteindre un résultat final. Le logiciel de Chainlink est caractérisé par une multitude d'adaptateurs de base, y compris les requêtes HTTP, l'analyse en format JSON et la conversion en divers formats Blockchain. Jusqu'à présent, Chainlink fournit 15 adaptateurs de base. Le travail le plus courant effectué par les nœuds Chainlink consiste à utiliser une requête GET avec une API externe, afin d'extraire les données requises, à les convertir en données compatibles avec Blockchain, puis à renvoyer la réponse via le contrat intelligent de Chainlink en chaîne pour le User Smart Contract.

- Adaptateurs externes : Au-delà des adaptateurs internes intégrés, Chainlink offre la possibilité de définir des sous-tâches personnalisées appelées adaptateurs externes. Ces adaptateurs sont des services externes, s'exécutant sur des serveurs hors chaîne, avec une API REST. En modélisant

les adaptateurs de manière orientée services, les programmes dans n'importe quel langage de programmation (Python, GO, NodeJS, etc.) peuvent être facilement implémentés, simplement en ajoutant une petite API intermédiaire dans le programme. De même, l'interaction avec des API complexes à plusieurs étapes peut être simplifiée en sous-tâches individuelles avec des paramètres. Ainsi, le développement d'un adaptateur externe augmente les capacités de personnalisation de Chainlink et ouvre des portes à un grand nombre d'applications et de domaines qu'il peut prendre en charge. Les adaptateurs externes sont également destinés à couvrir certaines limitations existantes dans les adaptateurs internes, telles que l'authentification API. En effet, la récupération de données à partir d'une API nécessite généralement une authentification avec une clé API. L'utilisation d'un adaptateur de base rendra la clé API disponible publiquement sur la Blockchain (codée en dur dans User Smart Contract en tant que paramètre). Les adaptateurs externes peuvent prendre en charge ce cas car l'exécution de l'adaptateur est maintenue hors chaîne. De plus, étant donné que l'adaptateur externe est constitué de morceaux de code personnalisés qui s'exécutent sur des serveurs hors chaîne, il est utile d'effectuer le maximum de calculs hors chaîne sur ces serveurs plutôt que d'effectuer tous les calculs en chaîne (dans le User Smart Contract).

Le mécanisme de fonctionnement de Chainlink est le suivant : un User Smart Contract lance une requête. La demande comprend un JobID qui identifie le travail à effectuer par le nœud. Le Job définit l'initiateur et l'adaptateur impliqués dans la demande. Ces deux composants sont des instructions qui décrivent en détail la manière de répondre à la requête. Les adaptateurs spécifient la source de données externe et d'autres tâches pour les données telles que l'analyse JSON, la comparaison, conversion, etc. Tandis que les initiateurs fixent les circonstances de la réponse à la demande, quand et comment répondre : périodiquement ou à une heure spécifiée, déclenchée via des requêtes Web ou d'autres API, en surveillant les requêtes issues de n'importe quelle adresse de contrat ou d'une adresse spécifique, etc. La requête est détectée par le contrat intelligent de Chainlink puis reçue par le Chainlink Core. Le nœud analyse le travail à effectuer et les tâches adéquates sont alors exécutées. Si la requête appelle un adaptateur interne, le nœud demandera directement des données de l'API externe. Sinon si le travail fait référence à un adaptateur externe, le nœud lui enverra une demande pour récupérer les données de l'API externe.

Une fois reçue, la réponse est ensuite renvoyée par le nœud Chainlink, via une transaction, sur la chaîne, à l'intention du User Smart Contract.

2.1.2 Chainlink : décentralisation

Pour que Chainlink assure la sécurité, la décentralisation doit obligatoirement être établie. Comme nous l'avons mentionné précédemment, Chainlink implémente une décentralisation à 2 niveaux : la décentralisation d'oracles et la décentralisation des sources de données. De toute évidence, même avec plusieurs sources de données, s'appuyer sur un seul oracle Chainlink n'est pas sécuritaire, car un seul oracle Chainlink n'est pas nécessairement fiable. C'est pourquoi, une architecture Chainlink centralisée, en termes d'oracles, ne peut garantir l'exactitude des données. L'approche Chainlink aborde principalement deux problèmes. Tout d'abord, il traite le risque de défaillance du nœud Chainlink, alors il garantit la tolérance aux pannes à travers le recours à un réseau d'oracles. Deuxièmement, il vérifie l'intégrité des données reçues de la source externe, il assure ainsi la détection des nœuds défectueux ou malveillants et la sécurité contre les attaques. De ce fait, le framework Chainlink sécurise les données demandées ainsi que l'entité qui transmettra les données en chaîne, c'est-à-dire le réseau des oracles.

2.1.2.1 Décentralisation des sources de données

Pour effectuer la distribution des sources de données, plusieurs sources de données doivent être invoquées par les oracles. Un seul oracle peut récupérer des données à partir d'une ou plusieurs sources de données $src1, src2, .., srck$. Il doit ensuite collecter les réponses $a1, a2, .., ak$. La dernière étape consiste à agréger toutes les réponses en une seule valeur $A = Agg(a1, a2, .., ak)$. Il existe de nombreuses façons d'effectuer l'agrégation de données. Il peut définir un nombre minimum de réponses (k/2 par exemple), une fois reçu, l'oracle peut effectuer l'agrégation et envoyer la réponse au User Smart Contract. La procédure d'agrégation peut inclure une analyse des données pour détecter par exemple des valeurs extrêmes ou douteuses afin de pouvoir les éliminer. L'agrégation dépend du type de données. Comme il s'exécute actuellement sur Ethereum, Chainlink prend en charge les types de données suivants : int256, uint256, bytes32

et bool. Par exemple, si l'oracle traite int ou uint, il peut agréger les données en calculant la médiane ou la moyenne.

2.1.2.2 Décentralisation des oracles

La principale mesure de sécurité dans Chainlink est la décentralisation d'oracles. En fait, au lieu d'impliquer un seul oracle, un User Smart Contract doit contacter une collection d'oracles $O1, O2, .., On$. Chacun d'eux peut demander des données à partir d'une ou plusieurs sources de données. Le choix du nombre d'oracles dépend du niveau de sécurité souhaité par l'utilisateur, et il est proportionnel à son coût en termes d'ETH et de LINK. Impliquer un groupe d'oracles pour obtenir automatiquement des données externes nécessite une méthode d'agrégation sécurisée. Il y a 3 manière d'effectuer l'agrégation. Tout d'abord, au sein du User Smart Contract. L'utilisateur peut initier des requêtes à différents oracles, une fois qu'il reçoit les réponses, il peut les agréger. Deuxièmement, le composant Chainlink en chaîne de l'ensemble du système (c'est-à-dire les contrats intelligents des oracles) peut prendre la responsabilité d'agréger les données. Cet aspect est seulement théorique et pas encore applicable pratiquement sur Chainlink. Alternativement, l'agrégation peut être détenue par un autre contrat dédié à l'agrégation. Le contrat d'agrégation reçoit toutes les réponses des oracles, les agrège en un seul résultat final et ce dernier sera reçu par User Smart Contract. La figure 2.2 présente un exemple d'une architecture Chainlink entièrement décentralisée. Oracle Contract désigne toute la partie en chaîne de Chainlink (c'est-à-dire l'ensemble complet des contrats intelligents des nœuds), $Nodei$ signifie la partie hors chaîne du nœud Chainlink i et les API externes présentent les sources de données externes. Dans cet exemple, Oracle Contract est alors responsable d'agréger les réponses en une seule réponse et l'envoyer au User Smart Contract.

2.1.3 Chainlink : réputation

Outre la décentralisation, Chainlink atteint la sécurité grâce à la réputation. Premièrement, Chainlink propose un système de validation qui surveille le comportement des oracles en mesurant les performances via un ensemble de métriques objectives. Cela se fait par le contrat

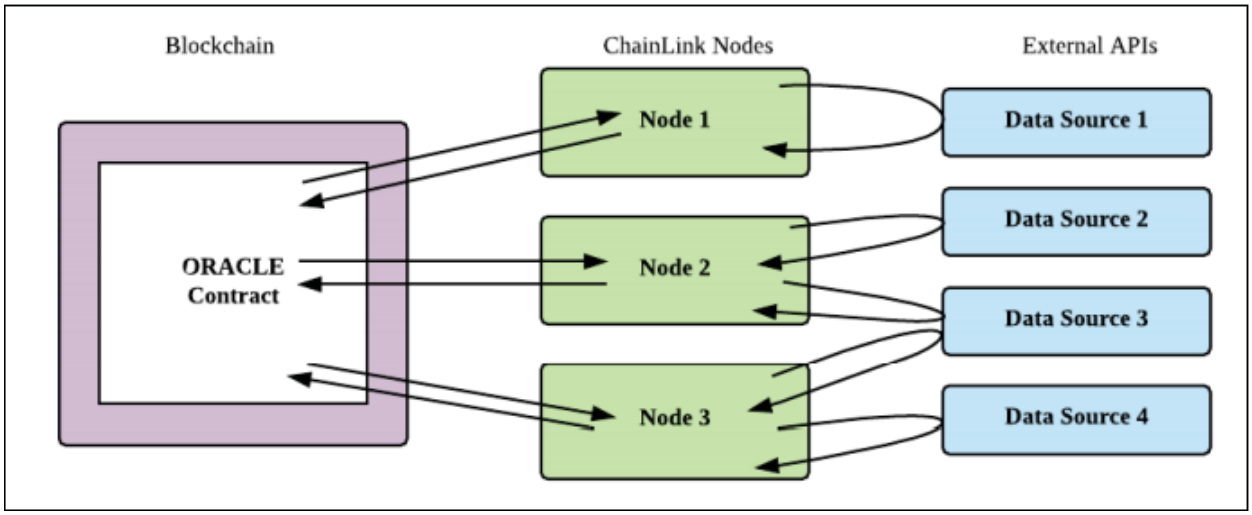

Figure 2.2 Un exemple d'architecture Chainlink entièrement décentralisée

de réputation. Il vise à fournir aux utilisateurs les attitudes des oracles en chaîne concernant leur disponibilités et leur exactitudes. Par exemple, Chainlink fournit des mesures de performance d'oracles telles que le nombre total de transactions, le total de LINK gagnés, le taux de réponse et le temps de réponse moyen de tous les temps de chaque oracle. Le système de validation ne peut pas être basé sur le réseau d'oracles lui même car les oracles ne peuvent pas effectuer la surveillance par eux-mêmes. Par contre, puisque les oracles doivent signer leurs réponses aux requêtes des utilisateurs, Chainlink a déployé un contrat intelligent chargé de collecter des statistiques sur les échecs d'oracles. Ils ont proposé aussi de récompenser les oracles pour avoir soumis des attestations d'inexactitude ou de non-disponibilité. Les statistiques d'oracles en chaîne et hors chaîne sont accessibles publiquement aux utilisateurs, en temps réel. Afin de renforcer le système de validation, Chainlink a mis en place un système de réputation qui permet aux utilisateurs eux-mêmes d'évaluer les oracles avec lesquels ils ont opéré pour répondre à leurs requêtes. Ainsi, Chainlink fournit aux utilisateurs un aperçu global des oracles existants et de leurs performances. De cette manière, le choix des oracles devient plus pratique pour les utilisateurs sur la base de statistiques et de témoins réels et objectifs.

2.2 Analyse de l'architecture et travail proposé

Dans cette section, nous décrivons notre plateforme d'évaluation pour Chainlink. Nous décrivons d'abord les expériences qu'on a fait pour évaluer les performances d'un seul nœud Chainlink. Nous procédons ensuite à la l'étude et présentation de notre architecture Chainlink décentralisée.

2.2.1 Motivation

En fait, un nœud Chainlink est essentiellement un nœud qui exécute le logiciel Chainlink sur un type de ressource choisi. Comme nous l'avons expliqué précédemment, le nœud Chainlink est divisé en une partie en chaîne (le contrat intelligent et le client Ethereum) et une partie hors chaîne (logiciel et adaptateurs externes) et communique avec des sources de données externes qui jouent un rôle principal dans son écosystème. Il s'agit en effet de l'entité élémentaire d'un réseau décentralisé d'oracles, construit pour transférer des données entre les contrats intelligents et le monde hors chaîne. Compte tenu de la diversité de ses composants, de ses méthodes de déploiement, des types et options de ses paramètres de configuration, nous avons décidé d'évaluer le comportement d'un nœud Chainlink concernant la variation d'une sélection de ses paramètres internes et éléments de son écosystème.

2.2.2 Configuration d'un seul nœud Chainlink

- Nœud Chainlink local VS basé sur le Cloud : Le logiciel Chainlink peut s'exécuter de manière fluide sur une machine locale ou sur des serveurs Cloud. Pour un opérateur de nœud Chainlink, le choix du type de ressource est critique car il peut vraisemblablement affecter ses performances qualitatives en termes de scalabilité, de disponibilité et de tolérance aux pannes, et ses performances quantitatives en termes de temps de réponse et de coût. C'est pourquoi, il a un impact crucial sur sa réputation dans le réseau de Chainlink. Par conséquent, ce choix influence directement les gains de l'opérateur Chainlink. Chaque nœud Chainlink calcule son temps de réponse, c'est le temps écoulé entre la détection de la requête en chaîne et la réception du reçu de transaction et des confirmations de blocs après le renvoi de la réponse en chaîne.

Pour cette raison, il est important de mener une étude comparative entre les nœuds Chainlink locaux et les nœuds Chainlink basés sur le les services Cloud. Pour les mesures de performances quantitatives, la principale mesure décisive du point de vue de l'utilisateur et du point de vue de l'opérateur de nœud est le temps de réponse et le coût du nœud Chainlink, car ils permettent d'évaluer les performances du nœud. C'est pourquoi nous nous concentrons principalement sur la comparaison des machines locales et Cloud en fonction de ces métriques.

- **Adaptateur interne VS externe :** Les adaptateurs sont une autre caractéristique importante qui joue un rôle clé dans Chainlink. Les adaptateurs sont définis comme l'outil utilisé par les nœuds pour récupérer des données à partir de sources externes. Alors que les adaptateurs internes sont des tâches nativement intégrées dans Chainlink, les adaptateurs externes s'exécutent séparément du logiciel natif Chainlink. Cependant, pour les tâches courantes pouvant être remplies à la fois par des adaptateurs des base et externes, une question se pose concernant le type d'adaptateur à choisir. Encore une fois, la principale mesure en termes de performances quantitatives qui peut décider le type d'adaptateur à choisir est le temps de réponse.

- **Paramètres logiciels et matériels de Chainlink :** Une fois la méthode de déploiement du nœud Chainlink est étudiée et après avoir étudié ses composants et connexions, nous avons décidé d'analyser en premier lieu ses paramètres matériels en termes de type de disque et de taille de mémoire. Il est important que les opérateurs de nœuds connaissent la capacité nécessaire et suffisante pour répondre aux requêtes sans être submergés. Deuxièmement, étant donné que l'ensemble du mécanisme Chainlink est lié à Blockchain, nous avons analysé ses performances en faisant varier les fournisseurs de clients Ethereum. Chaque fournisseur établit ses propres méthodes pour gérer l'équilibrage des ressources et son architecture, ainsi, les performances des clients Ethereum peuvent influencer celle du nœud Chainlink. Les initiateurs sont un autre élément clé du nœud Chainlink. En particulier, nous avons exploré Runlog et Cron. Runlog est un initiateur typique qui surveille les événements des journaux Blockchain pour un JobID spécifique. Cron, d'autre part, gère les tâches planifiées, en d'autres termes, le nœud Chainlink envoie périodiquement des données à un contrat intelligent spécifique sans qu'aucune demande initiale ne soit nécessaire. Le travail peut être lancé de manière autonome dans un intervalle de

secondes, minutes ou heures. Enfin, en élargissant la portée de l'étude à l'écosystème Chainlink, nous avons mené des expérimentations sur l'attitude des nœuds Chainlink envers différentes sources de données.

- Architecture centralisée de Chainlink : Afin d'évaluer de manière indépendante un nœud Chainlink en ce qui concerne sa méthode de déploiement, ses adaptateurs et un ensemble de ses paramètres logiciels et matériels à chaque fois, nous avons mis en place des systèmes Chainlink centralisés. Certes, la centralisation n'est pas notre objectif puisqu'un modèle centralisé est n'est pas sécuritaire et Chainlink est même basé sur la décentralisation. Cependant, nous nous concentrons dans cette étape sur le comportement d'un seul oracle dans l'écosystème Blockchain afin d'évaluer ses performances. L'architecture conçue est représentée par la figure 2.3. Un contrat intelligent demande des données externes à travers un nœud Chainlink qui utilisera un adaptateur pour récupérer les données requises. Selon le but de l'expérimentation, le nœud Chainlink peut être déployé localement ou à base de Cloud, l'adaptateur peut être interne ou un adaptateur externe déployé localement ou via une fonction Cloud Serverless. Enfin, nous avons choisi de travailler avec des API Web externes comme source de données externe. Pour les expériences relatives aux paramètres logiciels et matériels du nœud Chainlink, nous exécutons notre nœud Chainlink à base de services Cloud.

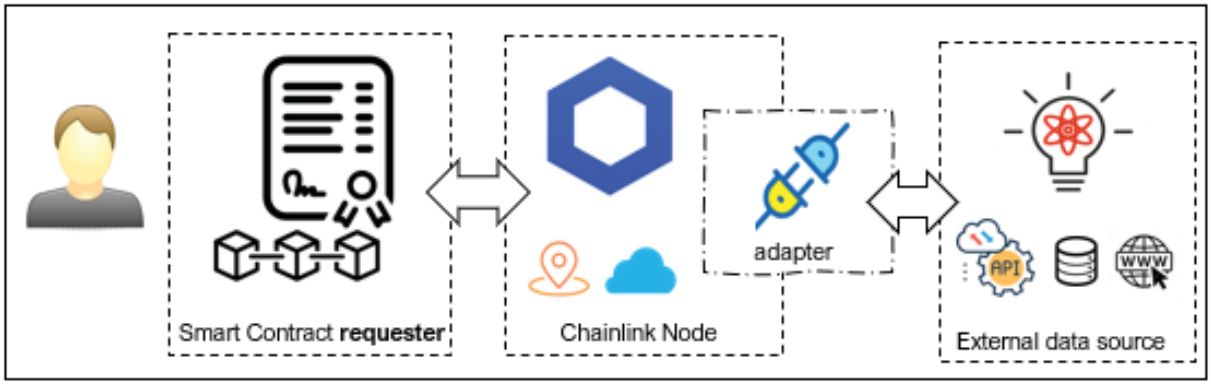

Figure 2.3 Conception de l'architecture centralisée

2.2.3 Étude d'un réseau d'oracles Chainlink

La conception et l'implémentation d'une architecture centralisée visaient principalement à étudier la configuration matérielle et logicielle des oracles Chainlink afin de déterminer les options optimales à suivre en tant qu'utilisateur Chainlink en général et opérateur de nœud en particulier. Cependant, un système Chainlink centralisé est certainement vulnérable car la sécurité globale est assurée par la décentralisation, en plus de la réputation. Nous avons alors conçu, mis en œuvre et évalué un système décentralisé basé sur Chainlink comprenant trois nœuds Chainlink et deux sources de données externes, illustrés par la figure 2.4. Un contrat intelligent demande des données aux nœuds Chainlink. Les nœuds Chainlink 1 et 2 demandent la même source de données tandis que le nœud Chainlink 3 demande une source de données différente. Ainsi, nous assurons la décentralisation d'oracles ainsi que la décentralisation des sources de données. Les deux sources de données sont des API de suivi IP qui, pour une adresse IP spécifiée, enverront un ensemble d'informations telles que le pays, la région, la ville, la latitude, la longitude, la devise, etc. Nous avons programmé les adaptateurs pour récupérer la latitude et la longitude de la réponse de l'API. les informations sont ensuite renvoyées en chaîne pour le contrat qui a lancé les requêtes.

Dans ce chapitre, nous avons présenté notre étude théorique qui présente le plan pour l'étude expérimentale de Chainlink. Nos investigations ciblent en premier lieu le composant principal de Chainlink : le nœud Chainlink. Il est ainsi important d'analyser son attitude face à la variation de ses paramètres, afin de choisir les meilleures options et d'être conscient de leurs impacts sur les performances du nœud. Pour cette raison, nous avons conçu une architecture centralisée dans laquelle un contrat intelligent demande des données à un nœud Chainlink, ce qui déclenchera un adaptateur interne ou externe pour récupérer les données. Une fois qu'un seul oracle est théoriquement étudié, nous avons conçu une architecture décentralisée, à deux niveaux, afin d'étudier et d'évaluer les résultats d'un réseau décentralisé d'oracles Chainlink.

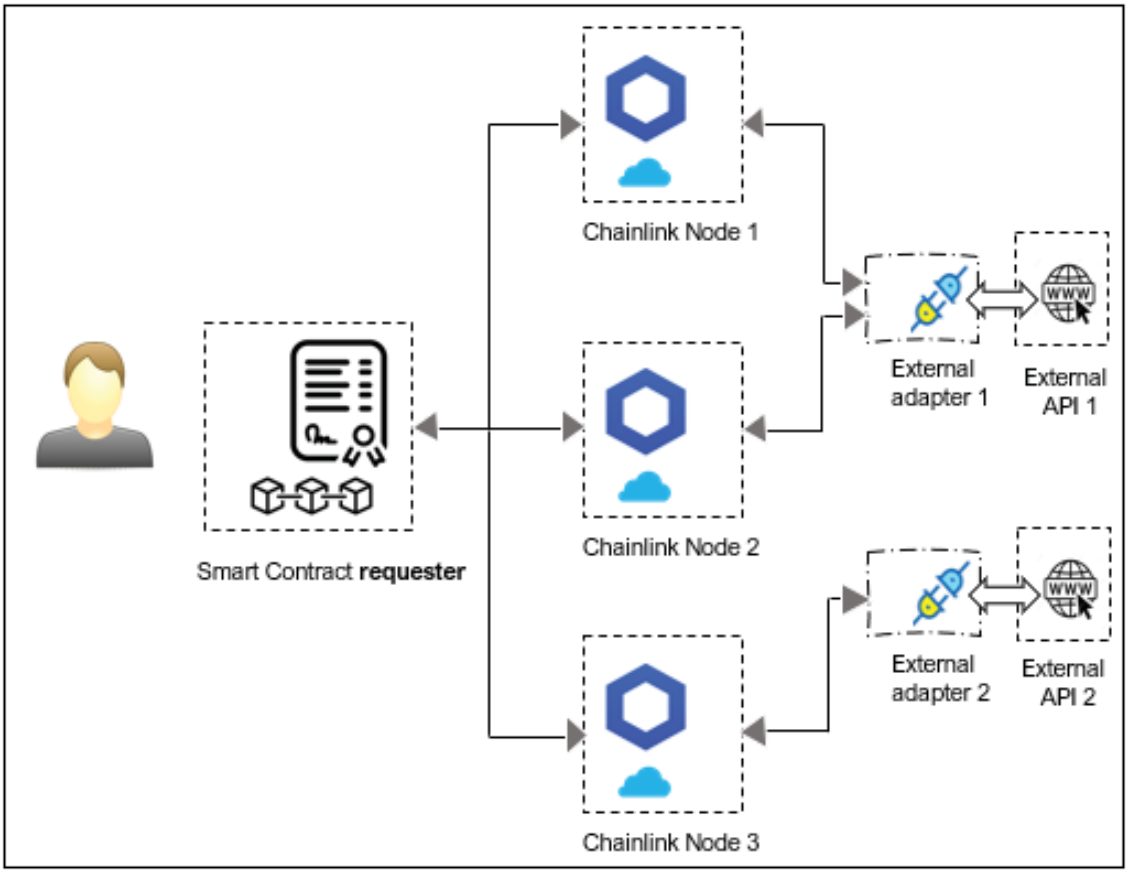

Figure 2.4 Conception de l'architecture décentralisée

CHAPITRE 3

RÉSULTATS ET ANALYSE DE PERFORMANCE

Dans ce dernier chapitre, nous commençons par décrire le montage de nos expériences et notre méthodologie. Deuxièmement, nous présentons les résultats de nos expériences. Dans un premier temps, nous présentons l'analyse des performances d'un nœud Chainlink à l'échelle matériel et logiciel. Ensuite, nous mettons en œuvre une architecture décentralisée où nous évaluons les performances globales du réseau Chainlink.

3.1 Mise en place et méthodologie expérimentale

3.1.1 Configuration d'un nœud Chainlink

Tant pour les nœuds locaux que pour les nœuds basés sur le Cloud, nos systèmes sont très proches en termes de caractéristiques matérielles : 8Go de RAM pour les machines locales et Cloud, stockage HDD pour les deux solutions et CPU de 2,5 Ghz pour la machine locale et de 2,7 Ghz pour la machine basée sur le Cloud.

Du point de vue logiciel, nous avons exécuté nos machines Chainlink à travers Docker. Outre le logiciel Chainlink, un serveur de base de données PostgreSQL est nécessaire pour stocker des informations sur les transactions et les circonstances globales d'exécution de Chainlink. De plus, comme le framework Chainlink ne peut être déployé que sur Ethereum pour le moment, un client Ethereum est fondamental. Cela peut être accompli en exécutant Geth, Parity ou un tiers comme Infura [8] qui fournit un point de terminaison pour un client Ethereum. En conséquence, nous avons configuré nos machines locales et virtuelles comme suit : nous avons exploité Infura pour le client Ethereum et nous avons utilisé PostgreSQL 12. Pour le nœud basé sur le Cloud, nous l'avons implémenté à travers les services Cloud de Google [9].

[8] Infura : https ://infura.io/

[9] Google Cloud Platform : https ://cloud.google.com/

3.1.2 Configuration des adaptateurs

Alors que les adaptateurs internes sont des services intégrés, les adaptateurs externes sont des outils auto-développés. Les adaptateurs externes sont aussi simples qu'un serveur écoutant aux demandes des clients et répondant aux requêtes. Il peut être implémenté en Python ou NodeJS. Pour le nœud local, nous avons exécuté le serveur localement sur Visual Studio Code, tandis que pour le nœud basé sur le Cloud, nous l'avons lancé via une fonction sans serveur.

Pour nos expériences avec des nœuds locaux et basés sur le Cloud, nous avons exécuté des serveurs NodeJS qui renvoient le prix de ETH en USD. Dans ce contexte, nous avons utilisé l'API cryptocompare [10].

Dans la figure 3.1, nous illustrons l'architecture que nous avons implémenté dans l'environnement de Google Cloud. Nous avons utilisé Google Compute Engine pour exécuter la machine virtuelle, un serveur Google PostgreSQL pour la base de données et Google Functions pour exécuter les adaptateurs externes à travers des fonctions sans serveur. Pour le système local, nous exécutons le logiciel Chainlink et PostgreSQL sur la même machine. La requête lancée par le contrat intelligent spécifie le travail à effectuer par le nœud Chainlink. Dans le premier flux de travail, un nœud Chainlink doit utiliser un de ses adaptateurs de base pour obtenir les données. Dans le deuxième flux, le nœud doit envoyer une requête à un adaptateur externe afin de récupérer les données.

3.1.3 Configuration du système décentralisé

Pour l'architecture décentralisée présentée précédemment, nous avons implémenté nos machines Chainlink sur des machines Google, configurées comme suit : mémoire de 3,75G, disque HDD, initiateur Runlog, fournisseur Ethereum Infura et adaptateurs externes déployés sur des fonctions sans serveur de Google. Les deux sources de données sont des API de suivi IP qui, pour une IP spécifiée, enverront un ensemble d'informations telles que le pays, la région, la ville, la latitude,

[10] Cryptocompare : https ://min-api.cryptocompare.com

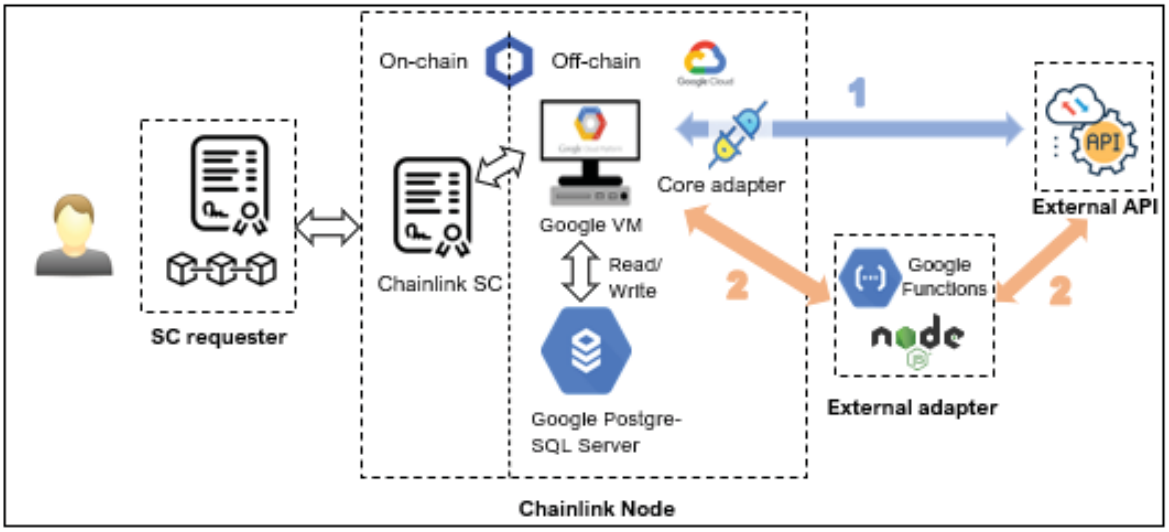

Figure 3.1 Système Chainlink centralisé basé sur les services Google

la longitude, la devise, etc. Nous avons développé les adaptateurs via NodeJS et les avons programmés pour récupérer la latitude et la longitude de la réponse de l'API. Les informations sont ensuite renvoyées en chaîne pour le contrat qui a fait la demande. Les API utilisées sont IPregistry [11] et IPwhois [12]. Pour évaluer l'ensemble du système, nous avons enregistré le temps de réponse pour chaque nœud Chainlink, les frais de transaction Ethereum en ETH, les frais de transaction Chainlink en LINK et les frais Cloud pour un total de 500 requêtes réparties en cinq intervalles. Pour toutes nos expériences, nous étions basés sur la plateforme Ethereum de Blockchain, car Chainlink fonctionne actuellement uniquement sur Ethereum. De plus, tous nos systèmes sont déployés sur le réseau Kovan de Ethereum.

3.1.4 Méthodologie

Pour analyser les performances des nœuds Chainlink, nous avons programmé un contrat intelligent pour lancer des requêtes aux nœuds Chainlink afin de récupérer des données. L'interface graphique de Chainlink fournit le temps de réponse pour chaque requête. Pour le temps de réponse des adaptateurs, nous avons pu l'extraire à partir des journaux de Chainlink.

[11] IPregistry : API de géolocalisation IP https ://ipregistry.co/

[12] IPwhois API de géolocalisation IP https ://ipwhois.io/

Le facteur temps est important dans ce contexte. Les circonstances du réseau d'Ethereum sont fortement dépendantes du temps. Ainsi, afin de bien mener nos expérimentations, et afin de minimiser efficacement la dépendance des résultats par rapport au temps, nous avons procédé selon la méthodologie suivante :

1. Pour l'analyse du comportement du nœud Chainlink par rapport à chaque paramètre, nous mettons en œuvre plusieurs systèmes centralisés en parallèle, toutes les configurations internes et externes sont parfaitement similaires à l'exception du paramètre que nous avons l'intention d'analyser. Afin de surveiller les performances des systèmes, nous lançons les requêtes consécutivement (une fois la requête du premier système est accomplie, nous lançons immédiatement la requête pour le deuxième système).
2. nous avons mené les expérimentations sur des intervalles de temps différents. L'intervalle de temps dans ce contexte est la différence temporelle entre un certain nombre de requêtes. Par exemple, nous lançons 10 requêtes, après 2 heures, nous initions à nouveau d'autres requêtes et après 17 heures nous renvoyons d'autres demandes.
3. A chaque intervalle de temps, nous varions le nombre de requêtes. Par exemple, dans l'intervalle de temps $T1$, nous envoyons 15 requêtes consécutives, en $T2$ nous envoyons 25 requêtes et en $T3$ nous envoyons 10 requêtes.
4. Pour améliorer l'indépendance temporelle, pour chaque intervalle nous faisons varier le temps entre les demandes d'une même solution. Par exemple, entre la demande 1 et 2, il existe un délai de 3 minutes, entre la demande 2 et 3, il ya un délai de 5 minutes et entre les demandes 3 et 4, un délai de 70 secondes.

Ces mesures aident à construire une analyse et évaluation qui ne dépendront pas des circonstances de la blockchain (exemple : congestion du réseau, prix du gaz en ETH), qui sont variables en fonction du temps.

3.2 Évaluation des oracles Chainlink

3.2.1 Impact de la méthode de déploiement d'un nœud Chainlink

Le choix de la méthode de déploiement du nœud Chainlink est un problème critique pour les opérateurs de Chainlink car il détermine le niveau de performance et a un impact considérable sur la réputation de l'opérateur dans le réseau Chainlink. Les figures 3.2 et 3.3 montrent respectivement la fonction de distribution cumulative (CDF) pour le temps de réponse des nœuds Chainlink locaux et basés sur le Cloud utilisant un adaptateur de base et un adaptateur externe. Pour ce faire, nous avons réalisé 800 requêtes pour chaque système.

Nous pouvons remarquer que pour les deux expériences, le nœud local est légèrement plus rapide à répondre aux requêtes que celui basé sur le Cloud. Ceci est attendu puisque les services Cloud nécessitent un délai supplémentaire et se sont limité au débit de l'Internet. Plus précisément, la machine Chainlink locale prend en moyenne 21,97 secondes pour répondre à une requête tandis qu'une machine basée sur le Cloud le fait en 22,76 secondes en moyenne. Les temps de réponse pour les deux types de machines Chainlink sont compris entre 13 et 37 secondes. Pour 55% des requêtes, le nœud local est 2,5 secondes plus rapide avec l'adaptateur de base et 3 secondes plus rapide avec un adaptateur externe. Statistiquement, pour le premier système (représenté par la figure 3.2), dans 60% des requêtes, le temps de réponse est inférieur à 22s en machine locale contre 52% pour la machine Cloud. Pour le deuxième système, 63% des requêtes ont un temps de réponse inférieur ou égal à 22s pour la machine locale contre 47% pour la machine basé sur le Cloud.

En ce qui concerne le coût des solutions, la machine locale ne coûte évidemment aucun frais continu, à l'exception de l'électricité, tandis que le nœud Chainlink basé sur le Cloud coûte environ 126,69 USD dans le cas où il fonctionne 24/7, avec le fournisseur Google. Comme pour un opérateur de nœud débutant, nous recommandons de commencer avec une machine Chainlink locale car c'est légèrement plus rapide, n'impose aucun frais supplémentaire et puisque le revenu est gagné à 100%, c'est considéré comme un début motivant. Une fois que l'opérateur

s'intègre profondément dans le réseau Chainlink et s'engage avec plusieurs utilisateurs et applications Chainlink, nous recommandons de migrer vers le Cloud, malgré que les machines locales soient plus rapides. En fait, la réputation de nœud Chainlink est accessible au public et affiche le taux de réponse. Il reflète le pourcentage de disponibilité du nœud pour répondre aux requêtes et une faible valeur donne une image non appréciée aux utilisateurs. De plus, selon la nature de l'application, de nombreux utilisateurs s'appuieraient primordialement sur le taux de réponse pour décider l'opérateur de nœud Chainlink à travailler avec. Par conséquent, la haute disponibilité et la scalabilité doivent être des caractéristiques clés dans les nœuds Chainlink. Dans ce contexte, les fournisseurs de Cloud Computing proposent ces services, difficiles à réaliser dans les systèmes locaux, de plus de la maintenance, des mises à niveau plus faciles et des ajustements matériels et logiciels flexibles.

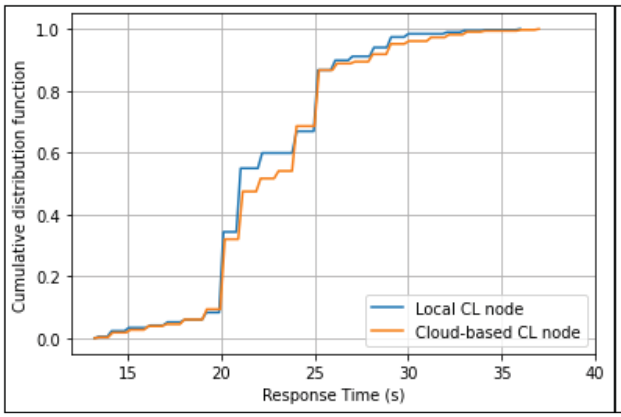

Figure 3.2 Temps de réponse avec un adaptateur interne

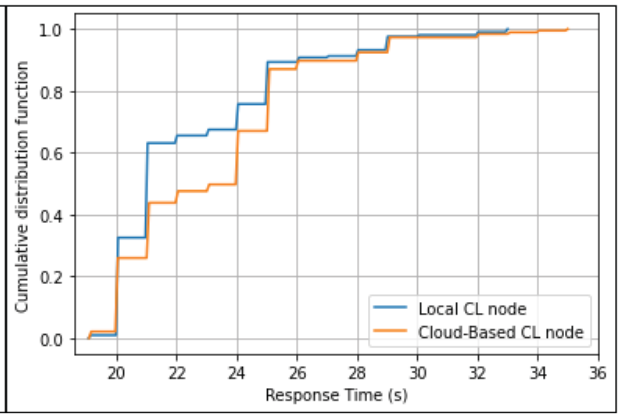

Figure 3.3 Temps de réponse avec un adaptateur externe

3.2.2 Impact du type d'adaptateur

Nous avons mentionné précédemment que pour un bon ensemble d'applications, les deux types d'adaptateurs, internes et externes, peuvent être appliqués. Ainsi, nous avons surveillé pour les deux systèmes, local et basé sur le Cloud, les temps de réponse des deux types d'adaptateurs vis à vis de la même source externe, l'API Cryptocompare. Pour ce faire, nous avons réalisé 800 requêtes pour chaque système. La figure 3.4 présente le CDF du temps de réponse pour les deux

adaptateurs dans un système local et la figure 3.5 correspond à celui basé sur le Cloud. Quant à leurs performances, nous pouvons remarquer que dans le système local, les temps de réponse pour 82% des requêtes sont presque identiques et se situent entre 0,475s et 1s. La différence est plus explicite dans le système basé sur le Cloud où l'adaptateur interne est en moyenne 0,3 s plus rapide. En effet, dans 84% des requêtes, le temps de réponse de l'adaptateur interne est inférieur à 0,63s tandis que pour 85% des requêtes, le temps de réponse est compris entre 0,67 s et 0,98 s. Ainsi, les adaptateurs internes sont légèrement plus performants, ils sont recommandés pour les tâches simples et fréquentes. Cependant, l'adaptateur externe devient plus intéressant pour les API plus complexes car il est entièrement personnalisable et interopérable avec tout type de source de données. En particulier, pour les fonctions sans serveur de Google, 2500 appels d'une fonction légère coûtent 2.33 USD par mois.

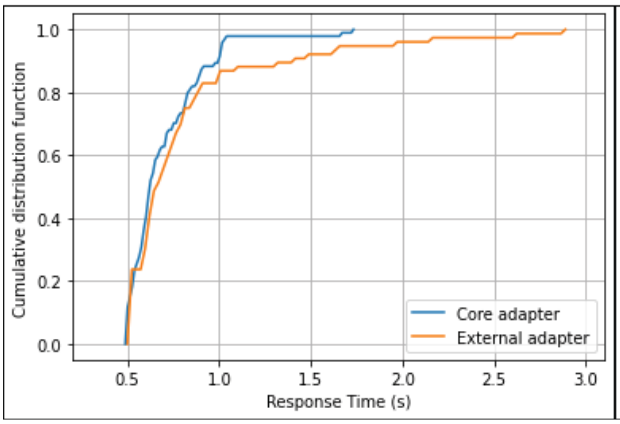

Figure 3.4 Temps de réponse dans un nœud local

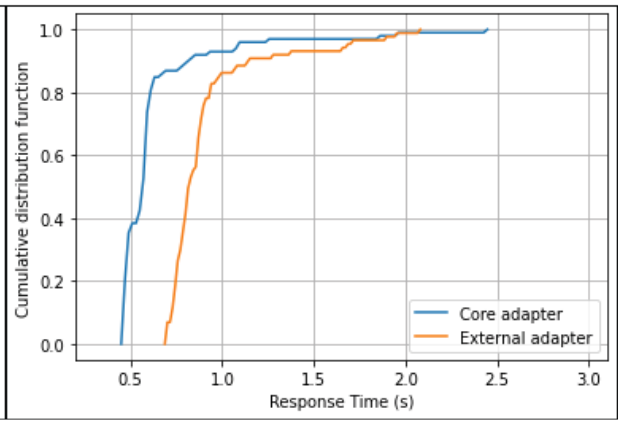

Figure 3.5 Temps de réponse dans un nœud basé sur le Cloud

3.2.3 Impact de la configuration matérielle

À l'échelle matériel, nous avons d'abord analysé les performances des nœuds Chainlink en fonction de la mémoire RAM. Nous avons testé trois possibilités via trois machines basé sur le Cloud, tous les paramètres sont identiques sauf pour la mémoire : 0.6G, 1.7G et 3.75G. Pour un exemple de 300 requêtes, nous avons enregistré le temps de réponse. Les résultats sont présentés sous forme de CDF dans la figure 3.6. Ici, les performances sont pratiquement uniformes.

La configuration de 3.75G fournit les meilleurs temps de réponse avec une légère différence par rapport aux autres solutions. Il est tout de même important de mentionner que pour une configuration 0.6G, la machine est plus susceptible de répondre après plus de 30 secondes, pour 0,2% contre 0,006% dans les autres configurations. C'est attendu pour une telle faible mémoire, par contre, la machine réussit toujours à accomplir les tâches sans être explicitement dépassée.

D'autre part, étant donné que le nœud Chainlink écrit activement des informations Blockchain dans la base de données PostgreSQL, nous avons comparé ses performances en fonction du type de disque, HDD et SSD, du serveur PostgeSQL. Les résultats, illustrés à la figure 3.7, démontrent que le temps de réponse n'est pas affecté par ce paramètre et l'influence est très minime.

Dans ce contexte, une mémoire de 3,75G avec un disque HDD sont adéquats. De plus, dans notre configuration matérielle, nous utilisons nos machines avec un processeur à 2,25 GHz, seulement 3% de la capacité du processeur a été utilisée.

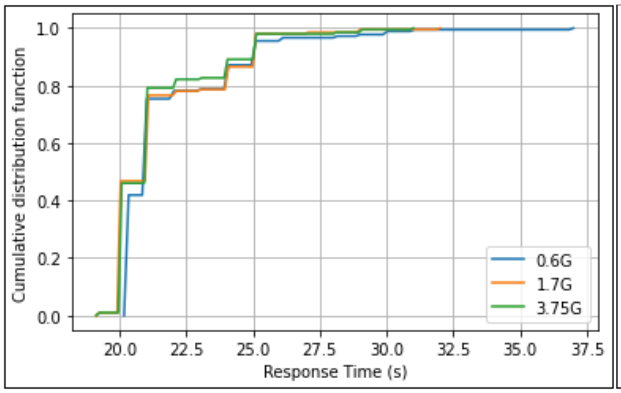

Figure 3.6 Impact de la mémoire

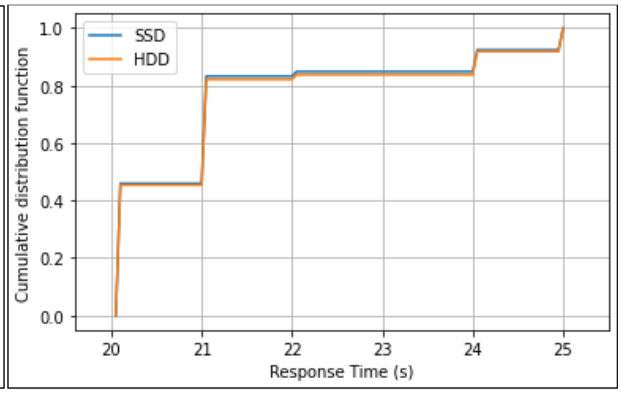

Figure 3.7 Impact du type de disque

3.2.4 Impact de l'initiateur

Outre la configuration matérielle, il existe un ensemble de paramètres Chainlink logiciels qu'un opérateur doit choisir. Dans ce paragraphe, nous présentons une comparaison entre les performances de deux initiateurs Chainlink : Runlog et Cron. La figure 3.8 évalue le temps de réponse des nœuds Chainlink pour chaque initiateur. En fait, les résultats démontrent que Cron a une influence explicite favorable. Les temps de réponse sont compris entre 13s et 17s où

89% d'entre eux sont égaux à 13s. Pour l'initiateur Runlog, le temps de réponse est compris entre 20s et 25s. Ceci est expliqué par le fait que la machine Chainlink gagne beaucoup de temps lorsqu'elle remplit sa tâche de manière programmée et automatique. De cette manière, elle n'a pas à surveiller puis analyser les journaux de Blockchain. En cas de tâches fréquentes et récurrentes, l'initiateur Cron est très convenable. En ce qui concerne les coûts de initiateurs, chaque opérateur décide, selon ses estimations, le coût final de son travail. Cependant, pour l'initiateur Cron, la tache se fait actuellement de manière gratuite, en attendant les mises à jour de la version du logiciel prochaine de Chainlink.

3.2.5 Impact du fournisseur Ethereum

Sur l'échelle Blockchain, nous avons évalué le temps de réponse en fonction du fournisseur Ethereum. Nous avons utilisé Infura, Quicknode [13] et Alchemy [14] dans des expérimentations de 300 requêtes. Ces services fournissent l'infrastructure de base d'Ethereum, aidant l'écosystème Blockchain à évoluer en nombreux de clients et en nombre applications décentralisées. La figure 3.9 montre l'impact de la variation des fournisseurs d'Ethereum. Comme nous pouvons le constater, les performances des solutions sont harmonieuses. Pour un temps de réponse inférieur à 22s, le système Quicknode est en tête avec 88% de requêtes, Alchemy 84% et Infura 78%. De plus, Quicknode a le moins de requêtes durant 24 secondes ou plus, 11% contre 17% pour Alchemy et 14% pour Infura. La différence de performance existe mais n'est pas profonde. Elle revient à la différence des architectures de chaque fournisseur Ethereum et leurs mécanismes d'équilibrage de charge. En ce qui concerne leurs tarification minimale, Infura propose la gestion gratuite de 100 000 requêtes Ethereum, Quicknode propose 300 000 requêtes/jour pour 9 USD et Alchemy propose 100 000 unités de calcul gratuites par mois. Dans ce cadre, Quicknode propose un rapport qualité-prix adéquat.

[13] Quicknode : https ://www.quicknode.com/

[14] Alchemy : https ://www.alchemy.com/

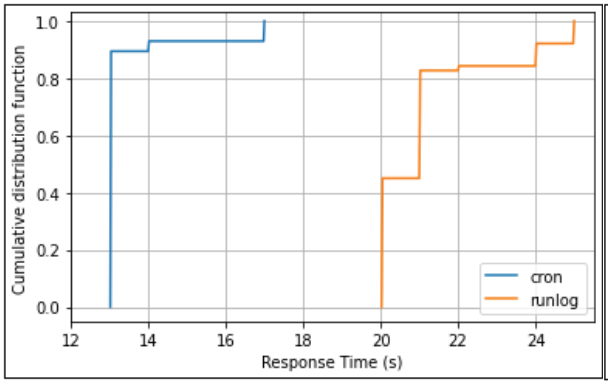

Figure 3.8 Impact de l'initiateur

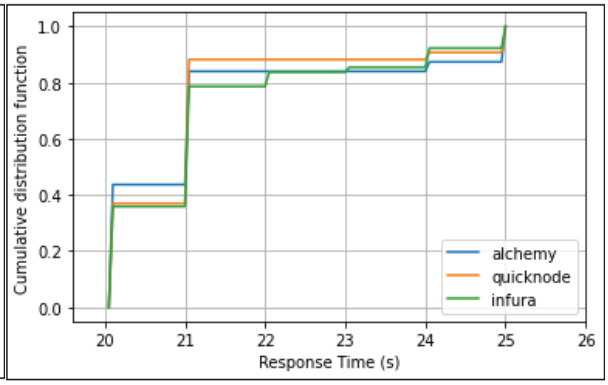

Figure 3.9 Impact d'Ethereum

3.2.6 Impact de la source de données

Étant donné que les oracles Chainlink sont principalement conçus pour récupérer des données du monde hors chaîne, ils ont un lien vital et permanent avec les sources de données externes. Dans ce contexte, les performances des oracles dépendent fortement du taux de réponse des sources de données. Ainsi, nous avons évalué le temps de réponse d'un nœud Chainlink sur la base de deux sources de données différentes, l'API cryptocompare et IPregistry. Pour un exemple de 300 requêtes, la figure 3.10 qui présente cette variation, prouve clairement une dissemblance notable dans les performances de l'oracle. Pour 80% des requêtes, il y a un écart de 4 secondes dans le temps de réponse. Pour l'API IPregistry, 89% des requêtes sont entre 20s et 21s tandis que 85% des requêtes sont entre 24s et 25s dans la solution Cryptocompare. Cette disparité est confirmé par la figure 3.11 qui illustre la variation du temps de réponse de chaque API. IPregistry répond dans 95% des demandes dans les 0,1 à 0,5 secondes, tandis que Cryptocompare répond à 60% des requêtes dans la plage de 3,5 et 6 secondes.

La conception et l'implémentation d'une architecture centralisée visaient principalement à analyser la configuration matérielle et logicielle des oracles Chainlink afin de déterminer les options optimales à suivre en tant qu'utilisateur Chainlink en général et opérateur de nœud en particulier. Elle nous a permis de suivre le comportement puis conclure concernant l'impact de ses paramètres sur ses performances. Certaines options ont notamment influencé le temps de réponse, pour une différence de 3 secondes et plus. Par contre, d'autres options ont introduit

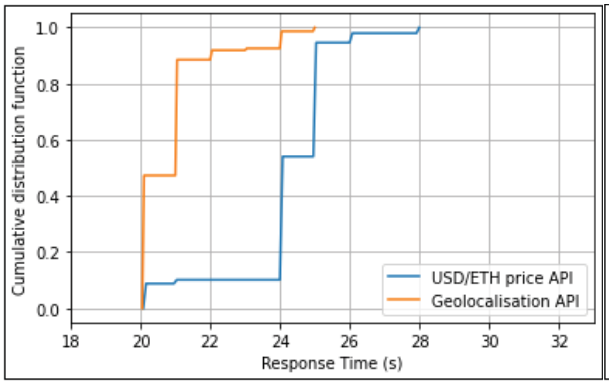

Figure 3.10 Impact de la source de données

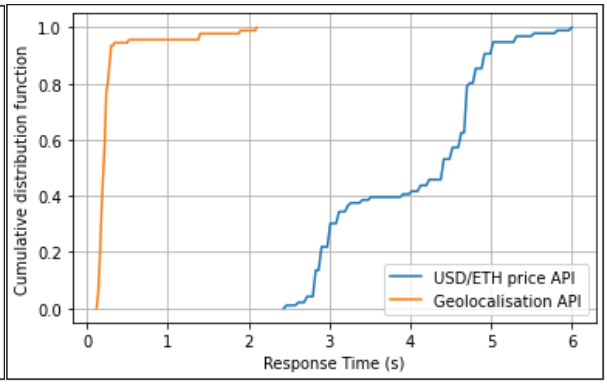

Figure 3.11 Variation du temps de réponse des sources de données

une différence inférieure à une seconde. Cette différence minime, cependant, pour un grand nombre de requêtes, pourrait provoquer un impact considérable. A titre d'exemple, dans le cadre de données en temps réel, pour 1000 requêtes par jour et une différence de 0.3 secondes entre l'adaptateur interne et externe, il y aurait un gap total de 300 secondes entre les deux systèmes par jour. Pour certaines applications telles que e-health, ce gap pourrait être critique.

Toutefois, un système Chainlink centralisé est certainement vulnérable car la sécurité globale est assurée par la décentralisation, en plus de la réputation. Dans le paragraphe suivant nous présentons les résultats d'un système décentralisé basé sur Chainlink.

3.2.7 Résultats d'une architecture décentralisée

Nous avons ainsi conçu, mis en œuvre et évalué un réseau d'oracles Chainlink comprenant trois nœuds Chainlink et deux sources de données externes, IPregistry et IPwhois, pour récupérer les données de latitude et de longitude et les soumettre en chaîne en faveur du contrat demandeur. Pour chaque 100 requêtes, nous avons enregistré et calculé le temps de réponse moyen pour chaque machine Chainlink, le coût de Ethereum, le coût Cloud et le coût LINK. Les mesures sont présentées dans le tableau 3.1. Comme nous l'avons analysé dans nos paragraphes précédents, le temps de réponse et le coût dépendent de plusieurs paramètres internes et externes. Pour cette architecture, nous avons déployé les nœuds sur des machines Google configurées comme

suit : mémoire 3.75G, disque HDD, initiateur Runlog, Infura comme fournisseur Ethereum et adaptateurs externes. Évidemment, le coût d'une telle architecture n'est pas statique car il dépend en permanence de la Blockchain (coût d'Éthereum). Comme le prix de l'Ether est activement variable, 100 requêtes peuvent coûter 0,1129 Ether comme elles peuvent coûter jusqu'à 0,4764 Ether et plus. Dans le réseau Mainnet Ethereum et selon les statistiques du prix de l'Ether à partir de septembre 2020 [15], le coût en terme d'Ethereum du système pour 100 requêtes coûtant 0,1129 Ether peut aller de 38,39 USD jusqu'à 471,80 USD. Concernant le coût de LINK, il est fixé par l'opérateur de nœud, c'est-à-dire que pour chaque travail qu'il propose, l'opérateur de nœud estime et décide du coût en LINK pour chaque tâche qu'il accomplit. Par exemple, nous avons configuré nos nœuds pour exécuter des tâches pour 0,3 LINK/requête. Pour chaque 100 requêtes, la moyenne des temps de réponse des nœuds est presque égale. En d'autres termes, dans les mêmes circonstances matérielles, logicielles et Blockchain, les nœuds Chainlink se comportent de manière harmonieuse et similaire. La variance commence à partir de 0,04 secondes et atteint 0,64 secondes. Le temps de réponse global moyen de ce système pour répondre à une requête d'un contrat est de 21,59 seconde/demande.

Tableau 3.1 Performance d'un réseau d'oracles Chainlink

Requêtes	**Moyenne Temps. Rep 1 (s)**	**Moyenne Temps. Rep 2 (s)**	**Moyenne Temps. Rep 3 (s)**	**Coût Ethereum (ETH)**	**Coût Cloud (USD)**	**Coût Chain-link (LINK)**
100	21.91	22.08	22.24	0.1263	11.05	30
100	22.07	21.66	21.43	0.2181	12.21	30
100	21.10	21.14	21.36	0.4764	10.85	30
100	21.25	21.45	21.59	0.1129	12.32	30
100	20.62	20.69	20.46	0.1830	11.52	30

Afin de mener des analyses plus approfondies concernant les performances du réseau, nous avons statistiquement illustré les temps de réponse dans le diagramme représenté par la figure 3.12. Comme nous pouvons le remarquer, 75% des demandes sont satisfaites en 20 à 21 secondes. Les temps de réponse sont compris entre 13s et 45s. Cependant, le nombre de requêtes qui

[15] https ://www.google.com/finance/quote/ETH-USD

sont à l'extrémité (inférieur à 20s ou supérieur à 31s) ne présentent que 4%. Cela reflète un comportement harmonieux des oracles Chainlink tout au long de la période des expérimentations. Nous notons que 93% des demandes sont satisfaites en 25s ou moins. Malheureusement, il n'y a pas de travaux de recherche antérieurs qui évaluent expérimentalement des réseaux Chainlink ou autres oracles comme Town Crier ou Provable, alors nous ne pouvons pas comparer les performances de Chainlink en termes de temps de réponse à d'autres solutions oracles. Cependant, les résultats prouvent que les performances de Chainlink sont pertinentes. Évidemment, le temps de réponse du système dans son ensemble dépend du mécanisme d'agrégation adopté pour recueillir et générer les résultats finaux. Par exemple, si un système de 10 oracles génère le résultat final après avoir reçu seulement 3 réponses, le temps de réponse du système sera probablement plus rapide que d'attendre les 10 réponses. D'un autre côté, la sécurité pourrait être moins solide et le coût poserait des problèmes, cela fait partie de nos travaux futurs, en continuant à explorer expérimentalement Chainlink au niveau de sa décentralisation.

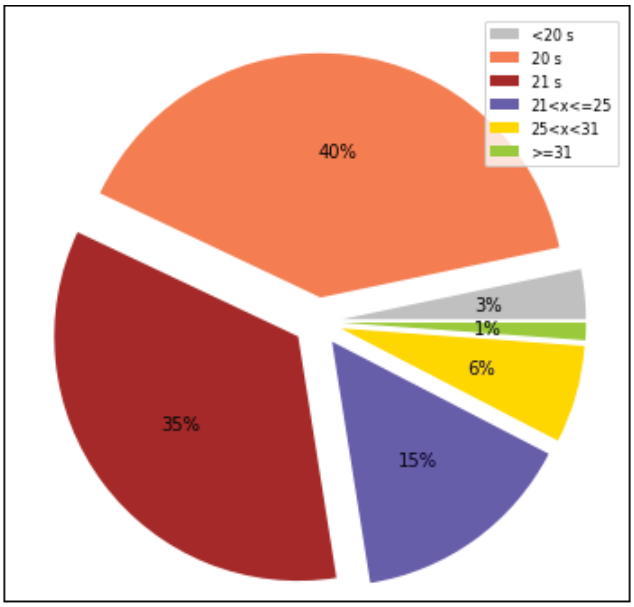

Figure 3.12 Pourcentage des temps de réponses

Dans ce dernier chapitre, nous avons commencé par présenter toutes les configurations de nos expérimentations : le nœud Chainlink, les adaptateurs et la configuration du réseau d'oracles. Nous avons également décrit la méthodologie que nous avons suivie lors des expérimentations.

Ensuite, nous avons présenté nos résultats, discuté et analysé les performances du comportement du nœud Chainlink vis à vis de ses paramètres matériels et logiciels. Enfin, nous avons évalué les performances d'un réseau d'oracles Chainlink en fonction son temps de réponse et de son coût total.

CONCLUSION ET RECOMMANDATIONS

Les oracles sont devenus un besoin essentiel pour alimenter les contrats intelligents avec les données externes. De multiples plateformes d'oracles ont été proposées afin de garantir la sécurité et d'éviter tout risque ou de menaces pour la Blockchain. Chainlink, une solution d'oracles récente et émergente en fait partie. Il s'agit en effet d'un réseau d'oracles décentralisé et basé sur la réputation, proposant de transmettre en toute sécurité des données à l'intérieur du système Blockchain pour les contrats intelligents. Dans ce travail de recherche, notre objectif principal est de couvrir l'étude expérimentale manquante de Chainlink. Nous avons ainsi analysé le comportement d'un seul nœud Chainlink en ce qui concerne la variation d'un ensemble de ses paramètres décisifs. Puis, nous avons évalué la performance globale du réseau décentralisé d'oracles Chainlink. Cela constitue la première évaluation expérimentale de Chainlink basée sur des systèmes mis en œuvre dédiés.

Tout d'abord, nous avons étudié la méthode de déploiement du nœud Chainlink. Les résultats montrent qu'un nœud local est plus rapide qu'un nœud basé sur le Cloud, en revanche, la machine Cloud est évidemment plus chère. En ce qui concerne les adaptateurs, les performances montrent que les adaptateurs de base sont légèrement plus rapides, cependant, les adaptateurs externes sont plus susceptibles de s'adapter à n'importe quelle application, ce qui couvre la limite des adaptateurs de base. À mesure que les opérateurs de nœuds approfondissent leurs bases dans la communauté Chainlink, nous recommandons de s'appuyer sur des adaptateurs externes et des machines basés sur le Cloud pour garantir la haute disponibilité, une bonne scalabilité, une maintenance et tolérance aux pannes plus facile et des ajustements plus flexibles. Ces détails influencent positivement la réputation des opérateurs.

En ce qui concerne la configuration matérielle, nous pouvons estimer qu'une machine de 3,75G de mémoire, de disque HDD et un CPU de 2,25Ghz est largement suffisante pour exécuter correctement et avec succès un nœud Chainlink. Nos évaluations prouvent également que

l'initiateur Cron est beaucoup plus efficace que Runlog. Cron est très favorable aux contrats intelligents qui demandent périodiquement des données externes. La source de données a également un impact pertinent sur les performances du nœud Chainlink. Aussi, nous avons testé différents fournisseurs de Client Ethereum, les résultats étaient quasiment identiques, le choix peut éventuellement dépendre du meilleur package proposé, dans ce contexte, nous recommandons d'utiliser Quicknode.

Au delà des caractéristiques élémentaires d'un nœud Chainlink, le réseau Chainlink décentralisé que nous avons mis en œuvre fournit aux utilisateurs et opérateurs de nœuds Chainlink une performance réelle d'un système basé sur Chainlink, déployé sur Ethereum, en termes de temps de réponse et de coût. Le temps de réponse moyen est de l'ordre de 21 à 22 secondes. Quant au coût du système, il varie fortement en fonction du prix de l'ETH, qui est à son tour variable et de plus en plus cher.

Comme peu de recherches ont étudié les oracles et qu'aucun travail précédent n'a étudié expérimentalement les composants et systèmes Chainlink, ce domaine est toujours ouvert à de nouvelles contributions et propositions. Premièrement, étant donné que dans le système Chainlink, plus il y a de nœuds et de sources de données, plus la sécurité est garantie, un mécanisme d'optimisation est alors obligatoire pour assurer l'équilibre entre la sécurité du système, la latence et le coût concernant le nombre de nœuds et de sources de données à activer. Cela appartient en fait à la technique d'agrégation à mettre en œuvre, et c'est dans le cadre de nos travaux futurs. D'autre part, une autre étude de recherche qui nous intéresse concerne la capacité et l'efficacité de Chainlink à couvrir le domaine de l'Internet des objets. En effet, l'IoT est une source importante de données. Alors que les contrats intelligents attirent l'attention de divers domaines, l'IoT ne fait pas exception. Dans ce contexte, une question peut être soulevée concernant la capacité et l'efficacité de Chainlink à gèrer les données IoT en temps réel en termes de fiabilité, latence et de coût.

ANNEXE I

ARTICLE SOUMIS

L'article intitulé "Benchmarking the Chainlink Oracle Network for Blockchains" est soumis à ACM SAC 2022 DAPP, The 37th ACM Symposium on Applied Computing (du 25 avril au 29 avril 2022 à Brno, République Tchèque).

Benchmarking the Chainlink Oracle Network for Blockchains

ABSTRACT

The emergence of smart contracts have allowed blockchain systems to cover applications beyond cryptocurrencies, such as Internet of Things, supply chains, healthcare, etc. In these modern applications, smart contracts will require data pulled from the real world. Consequently, oracles were proposed as a mechanism to feed external data to the blockchain in order to be read during smart contract execution. Chainlink is a popular example of a decentralized network. It is a highly configurable system which can gather data from a variety of sources, process them through multiple processing nodes (called Chainlink nodes) by running different types of adapters before storing them on-chain. However, Chainlink has not yet been thoroughly evaluated to understand the impact of various configuration parameters on the performance of the system. In this paper, we seek to address this gap by benchmarking Chainlink and analysing our results. We investigate the best options for node operators and users to adopt in their applications. In particular, we demonstrate how Chainlink nodes behave when varying a set of its internal (e.g., choice of adapter) and external parameters (e.g., hardware used). Furthermore, we implemented a decentralized model using cloud services and evaluated the overall performance in terms of response time and total cost of cloud, Ethereum and Chainlink services.

CCS CONCEPTS

• **Software and its engineering** → **Software architectures**;

KEYWORDS

Blockchain, Oracles, Chainlink, Smart contract

ACM Reference Format:
. 2022. Benchmarking the Chainlink Oracle Network, for Blockchains, . In *Proceedings of ACM SAC Conference (SAC'22).* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/xx.xxx/xxx_x

1 INTRODUCTION

The remarkable rise of blockchain technology in domains beyond cryptocurrencies can be attributed to the development of smart contracts that were introduced by Ethereum [1]. On its own, smart contracts are nevertheless limited in their functionalities as they can only read and write information that is managed on the blockchain itself. However, many potential applications require data from external sources, such as IoT sensors, physical documents, or even web sites [2].

To address these limitations, oracles have been proposed as a complimentary solution to smart contracts. They can be considered as third-party entities that link the blockchain system (on-chain data) to the outside world (off-chain data) [4].

Despite its essential benefits to smart contracts, introducing a single middleware between a blockchain and an external entity is a potential weakness, as it can introduce untrusted or malicious inputs to the blockchain. To address this security issue, several approaches have been proposed. For instance, oracles such as Provable employ hardware security technologies (e.g., Intel SGX) for reliable oracle execution [**?**].

In this paper, we focus on Chainlink, which is the most popular blockchain oracle service. Chainlink is an emerging, continuously evolving solution of oracles, launched in May 2019. Chainlink is based on decentralization and reputation in order to maintain security when delivering data to smart contracts [5]. A decentralized Chainlink-based system is a network of nodes that utilizes adapters to retrieve external data. Analyzing oracles performance is fundamental with a view to decide the best configuration to opt for to gain the optimal efficiency [6].

In this paper, we provide the following contributions:

(1) We provide a comprehensive study of Chainlink architecture, mechanisms and characteristics in detail: its principle, components and services.
(2) We experimentally evaluate the performance of a single Chainlink oracle with respect to internal and external parameters (hardware and software).
(3) We design and implement a testbed for evaluating a decentralized network of Chainlink oracles using cloud services. We analyze the overall performance in terms of response time and total cost with respect to the cloud, Ethereum and Chainlink services.

To the best of our knowledge, this is the first research work that is entirely dedicated to experimentally study Chainlink based on real system implementations, and present an evaluation of a network of Chainlink oracles.

This paper is structured in this manner: in Section 2, we study properly the concept of oracles and the leading approaches that exist up to date. In Section 3, we discuss Chainlink in details : its principle, architecture, components and services. In Section 4, we present our testbed to evaluate Chainlink on the centralized then decentralized scale. Section 5 presents our experimental study of Chainlink followed with discussions. Finally, Section 6 concludes our paper.

SAC'22, April 25 –April 29, 2022, Brno, Czech Republic

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/xx.xxx/xxx_x

2 BACKGROUND

In this section, we first describe the purpose and general mechanism of oracles. We also characterize them based on their centralized/decentralized architecture and inbound/outbound data flow direction. In the second part, we present the leading oracle solutions and their diversified architectures.

2.1 Oracles

Trust is a key feature in business dealings. Each storage system sets up its own means of security, privacy and trust. Blockchain, based on decentralization, ensures security thanks to its solid consensus and cryptographic algorithms popularized on its entire network. Oracles, a newly involved component in the blockchain ecosystem, is automatically required to robustly maintain the high security level afforded by the blockchain distributed system [7]. An oracle is a linker that connects smart contracts to the external world [8] . It is not a part of smart contracts nor it is the external information itself. It is a complete different component that transmits data between the requester and data source. The required data can be of multiple types: transaction information, real-time data, sensor data, etc. Also, a plenty of data sources are supported: Websites, databases, any API, etc.

Considering the significant need for oracles to feed smart contracts with external data, there is a dynamic trend that led to the implementation of many oracle solutions recently. This is why multiple approaches were proposed. According to [4], oracles can be categorized depending on several criteria: data source type (software,hardware or human), information circulation direction (inbound or outbound) and finally depending on its architecture (centralized or decentralized). In this work, we will focus two main categories: centralized/decentralized-based architecture and inbound/outbound oracles.

Centralized vs. Decentralized: Potentially expanding the scope of smart contract applications, blockchain users, especially business companies, are concerned about the oracle architecture. In other words, the architecture principle of oracles is a key feature that plays an important role in defining the system security level.

A centralized oracle is a one-component system that is responsible of linking smart contracts to the outside world [9]. All oracle tasks are controlled, managed and fulfilled by this single entity. Each solution proposes a different means of security. In particular, Provable is based on a set of authenticity proofs and TownCrier runs on Trusted Execution Environments (TEE). This centralization, despite its attempts to set up a good level of trust against attacks, it still immediately raises issues regarding its security. In fact, relying on a single component to incorporate data into the blockchain network can expose the blockchain system to breakdowns and failure. This will noticeably decrease the system's fault tolerance and high availability. Moreover, one malignant control over this single entity is sufficient to exhibit the whole blockchain-based system to real threats. Furthermore, including a single trustful-based security system (a centralized oracle) in blockchain ecosystem is contradictory to trustlessness, that is a fundamental concept in blockchain. It is also inevitable to mention that centralized oracles attempt mainly to ensure the integrity of data in its 3 different states: stored (inside the oracle), in computation process (inside the oracle) and in transmission (within the network: from the oracle to the smart contract). However, this mechanism assumes that the received data from the data source is fully reliable and valid to use. Thus, centralized oracles don't afford the ability to verify and ensure the trust of the data source information itself. This is, nevertheless, one of decentralized oracles features, to ensure the security of transmitting data to smart contracts as well as verifying data source information correctness.

In fact, the concept of decentralization within oracles is similar to the blockchain basis. A decentralized oracle ledger is a compound of oracles that is responsible each of obtaining the required information and transmit it to the blockchain [10]. Thus, the core theory behind this concept is indeed to focus on providing information through numerous non trusted entities rather than concentrating on a single trusted entity. This practice reinforces information integrity and exactness. There are many decentralized oracle platforms, such as Chainlink and Witnet that are also reputation-based solutions. It is important to mention that for Chainlink in particular, besides to oracles decentralization, one single oracle has the ability to request information from different data sources. In other words, Chainlink defines two levels of decentralization: oracle decentralization as well as data source decentralization. This is one of the reasons why Chainlink is now an emerging oracle dedicated to securely transmit real-world information to smart contracts based on 2 levels decentralization. Besides, another technique mechanism used by decentralized oracle solutions to enhance security, that is reputation.

The mechanism of reputation in reputation-based oracles operate in a way that oracles are chosen to fulfill clients requests depending on a set of reliable metrics that present their performance. Some oracle providers give the possibility to clients to choose the oracle they want to work with while other oracle providers distribute themselves requests between oracles. In both ways, the oracle selection highly depends on its reputation. Such a system incites oracle operators to provide the best service to users. Depending on the oracle provider, many metrics are took into account: highest response ratio, total number of transactions, lowest average response seconds, etc.

Inbound vs. Outbound: When deciding the oracle provider to work with, it is essential to define the needed information direction: is it from on-chain to off-chain or from off-chain to on-chain or both. Up to now in this paper, we put the accent on one direction only: from the outside world right to the smart contract. The oracle fulfilling this task is an inbound oracle, that sends data inside the blockchain. An example to this is an oracle that provides real-time retail tracking information to a smart contract [3].

Otherwise, outbound oracles are intended for transmitting data from smart contracts to the external world. An example to outbound oracle is one that sends funds deposition success to an e-learning website for subscription purposes.

Figure 1 illustrates an overview of the job of oracles inside the blockchain ecosystem. It presents the 2 existing architectures of oracles: centralized and decentralized.

Both directions expand the scope of smart contracts. They both double the domains that smart contracts can integrate and heighten the adaptability of smart contracts towards the market evolution.

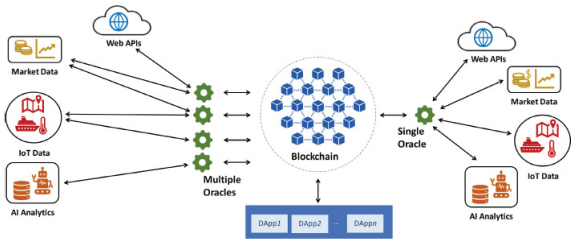

Figure 1: Oracles within the blockchain ecosystem

2.2 Oracle solutions

Provable: Operating since 2015, Provable[1] belongs to the family of centralized oracles. It can operate with a multitude of blockchain platforms like Ethereum, Hyperledger Fabric and Rootstock. it relies mainly on authenticity proofs to guarantee data integrity. In fact, the solution developed by Provable ensures the non falsification and the authenticity of the data retrieved from the external data source. This is accomplished by generating a document called authenticity proof and return it back with the response to the smart contract. Different technologies can handle the authenticity proofs, notably Trusted Execution Environments. It is based on the "If This Then That" concept. In other words, if a set of conditions are fulfilled, Provable will perform a set of actions. This helps the solution go beyond blockchain, to cover also other general contexts. To validate requests to Provable, the requester must include the data source type, the query and the authenticity proofs type.

Town Crier: Belonging to the centralized oracles, Town Crier (TC) establishes trust within oracles through the use of the Trusted Execution Environment Intel SGX instructions set, that is a new capability in certain Intel CPUs. Smart contracts request external data by the means of Town Crier from external data sources, more specifically, HTTPS-enabled internet services. Assuming that a client trusts SGX, TC garuantees that data transmitted from a website to a smart contract is absolutely not altered. In other words, it takes only to trust Intel's SGX as well as the data source, to be able to trust Town Crier. There is no need to trust the operators of TC or other party [10].

Witnet: Contrary to the previous oracle solution, the Witnet protocol exploits decentralization in order to connect smart contracts to external data sources. The protocol presents a distributed network of nodes, called witnesses. It has created its own token, Wit, with which Witnet rewards its oracles for retrieving external online data to user smart contracts [11]. In order to safely transmit data into the blockchain, a good number of witnesses, randomly selected, should be involved in order to retrieve data from one or more web sources. This can lead to form a trusted response if the majority of nodes have reliably reported data, through a consensus mechanism that can detect incoherence. To incite nodes correctly and faithfully transmit data into the blockchain, Witnet actually assigns jobs to witnesses in proportion to their behaviour in the network

Multiple interesting oracle propositions exist in the market. Each of them is distinguished by its own security measures and trust level. Despite its remarkable security level using Trust Execution Environments and even more security manners, the incorporation of a centralized party within the blockchain ecosystem contradicts the blockchain basis in terms of trustlessness and decentralization and exposes the blockchain system to the single point of failure problem. Accordingly, we guided our research to study decentralized oracles, more particularly, we are interested to study and analyze Chainlink, a new emerging and currently evolving solution for decentralized oracles. Therefore, in the next section we will outline this solution, its architecture, different components and characteristics.

[1]Provable documentation: https://docs.provable.xyz/

3 CHAINLINK: ARCHITECTURE AND CHARACTERISTICS

In this section, we thoroughly study Chainlink properties. First, we describe its components on the on-chain and off-chain scale, mainly: the Chainlink node and adapters. We also analyze its decentralization feature, on both layers, and finally present an overview about its reputation mechanism.

3.1 Overview

Defined as a decentralized oracle framework, Chainlink is a public open-source proposition to provide a great smart contracts need, that is external information. Holding a network of independent entities, the Chainlink framework ensures data retrieval, aggregation and dispatch to the smart contract requester. Chainlink was created in September 2017 and officially launched on Ethereum Mainnet in May 2019. The team designed Chainlink in a way that it supports different blockchain platforms. For now, it is built on and compatible with Ethereum. It created its own token, LINK, that allows Chainlink oracles to be paid for transmitting external data to smart contracts. Linking on-chain and off-chain environments, Chainlink has been designed with a high level of modularity. Every piece of the Chainlink system is upgradable, so that different components can be replaced as better techniques and competing implementations arise. It is also interesting to note that Chainlink team, in their long-term strategy, aims to integrate Trusted Execution Environment to their security measures and thus, combining centralized techniques to their decentralized solution [12].

3.2 Architecture

Creating a bridge between on-chain and off-chain, Chainlink architecture has two main sides, on-chain architecture and off-chain architecture. The on-chain side is responsible for interacting with the smart contract requester (user smart contract) and responding to the request (ie. detecting the request, then sending back the response on-chain to the requester). While the off-chain side is dedicated to fulfill the request (ie. process the request: connect to the external source and retrieve the concerned data). Based on this, many fundamental Chainlink nodes components are involved:

Chainlink-Smart Contract: This component belongs to the blockchain environment. Each Chainlink node runs a smart contract (Chainlink-SC) that listens to the blockchain logs in order to detects events issued by smart contracts that are in the charge of the Chainlink Node (User-SC). Once an event is detected, the Chainlink Core component receives the appropriate information to performs the required work.

Chainlink Core: The Chainlink core software is responsible for interfacing with the Chainlink-SC, scheduling, and balancing work across its various external services. Work done by Chainlink nodes is formatted as assignments. Each assignment is a set of smaller job specifications, known as subtasks (core adapters), which are processed as a pipeline. Each subtask has a specific operation it has to perform, before passing its result onto the next subtask, and ultimately reaching a final result. Chainlink's node software comes with a multitude of subtasks built in, including HTTP requests, JSON parsing, and conversion to various blockchain formats. Up to now, Chainlink provides 15 core adapters. The most common job performed by Chainlink nodes is to send a GET request to an external API, extract required data, convert it to blockchain compatible data, then submit the response back through Chainlink-SC right to the User-SC.

External adapters: Beyond the built-in Core adapters, Chainlink offers the possibility to define and develop customized subtasks known as external adapters. These adapters are external services, running on off-chain servers, with a minimal REST API. By modeling adapters in a service-oriented manner, programs in any programming language (Python, GO, NodeJS, etc.) can be easily implemented simply by adding a small intermediate API in the program. Similarly, interacting with complicated multi-step APIs can be simplified to individual subtasks with parameters. Thus, developing external adapter increases Chainlink customization capabilities and open doors to a large number of applications and domains that it can support. External adapters (EA) are intended also to cover certain limitations existing in core adapters such as API authentication. Retrieving data from an API generally requires authentication with an API key. Using a core adapter will turn the API Key be publicly available on the blockchain (hard coded in the User-Smart contract as a parameter). External adapters can handle this case as the execution of the adapter will be held off-chain.

The overview centralized architecture is illustrated in Figure 2. the Chainlink workflow is as follow: a User-SC initiates a request. The request includes a JobID which identifies the job to perform by the node. A job defines the initiator and adapter involved in the request. Both of these components are instructions that describe in details the manner of fulfilling the request. Adapters specify the external data source and other tasks for data like parsing, comparing, etc. While initiators fix the circumstances of the responding to the request, when and how to fulfill the request: periodically or at a specified time, getting triggered via web requests or other APIs, watching for requests from any smart contract address or from a specific one, etc. The request is detected by the Chainlink-SC and then received by the Chainlink Core. The node analyses the job to do and the adequate tasks are then performed. If the request calls for a core adapter then it will directly request data from the external API. Else if the job refers to an external adapter, then the node will send it a request to retrieve data from the external API. Once received, the response is then submitted back by the Chainlink node, via a transaction, on-Chain.

3.3 Decentralization

For Chainlink to be able to ensure security, decentralization must be established. As we mentioned earlier, Chainlink implements a

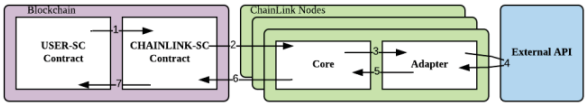

Figure 2: Overview of Chainlink centralized architecture

2-level decentralization: oracle decentralization and data source decentralization. Obviously, even with multiple data sources, relying on one Chainlink oracle is not at all secure, as a single Chainlink oracle is not trustful. Therefore, a centralized Chainlink architecture, in terms of oracles, cannot guarantee data correctness. The Chainlink approach addresses principally two issues. First, it deals with the risk of node failure, so it guarantees fault tolerance through engaging multiple nodes. Second, it verifies the integrity of data received from the external source, so it ensures the detection of faulty or malicious nodes and security against attacks. Thus, the Chainlink framework secures the required data as well as the entity that will transmit data on-chain, that is the network of oracles.

Data source decentralization: To perform data source distribution, multiple data sources must be invoked by oracles. A single oracle can retrieve data from one or multiple data sources $src1, src2, .., srck$. It then should collect the answers $a1, a2, .., ak$. The last step is to aggregate all the responses into one single value $A = Agg(a1, a2, .., ak)$. There are many ways to perform data aggregation. It can define a minimum number of answers ($k/2$ for example), once received, the oracle can perform the aggregation and send the response to the User-SC. The aggregation procedure can include data analysis to detect for example extreme or doubtful values so it can eliminate it. The aggregation depends on the data type. Since it is currently running on Ethereum, Chainlink supports the following data types: int256, uint256, bytes32 and bool. For example, if the oracle deals with int or uint, it can aggregate data through calculating the median or average.

Oracle decentralization: The main security measure in Chainlink is oracle decentralization. In fact, instead of requesting a single oracle, a User-SC has to contact a collection of oracles O1, O2, .., On. Each of them can request data from one or many data sources. The choice of the number of oracles depends on the system's needed security level, and it is proportional to its cost in terms of ETH and LINK. Involving a group of oracles to get external data automatically requires a secure method of aggregation. There are 3 components where aggregation can be performed. First, within the User-SC. The user can initiate requests to different oracles, once he receives the response, he can aggregate the answers. Second, the on-chain Chainlink component of the whole system (i.e, the oracles smart contracts) can take the responsibility of aggregating data. Alternatively, the aggregation may be held by another contract dedicated for aggregation. The final response will be then forwarded to the User-SC. Figure 3 shows an example of a fully decentralized Chainlink architecture. Oracle Contract denotes all the on-chain part of Chainlink (i.e, the full set of oracles smart contracts) , Node i signifies the off-chain part of the Chainlink oracle i and External APIs presents the external data sources. The on-chain component is then supposed to aggregate the answers into one single response and send it to the User-SC.

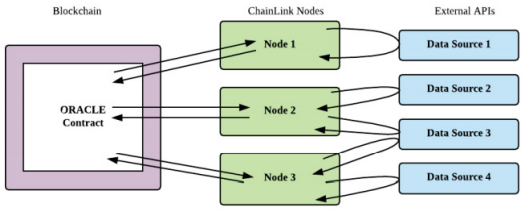

Figure 3: Example of a decentralized Chainlink architecture

3.4 Chainlink reputation

Besides to decentralization, Chainlink achieves security through its reputation-based characteristic. First, Chainlink offers a validation system that monitors oracles behavior through the measurement of performance via a set of objective metrics. It aims to provide for users oracles attitudes regarding their availability and correctness. For example, Chainlink provides oracles performance measurements like total transactions, total LINK earned, response ratio and all time average response time. The validation system cannot rely on the oracle network as oracles cannot perform monitoring by themselves. Otherwise, since oracles have to sign their responses to users requests, Chainlink deployed a smart contract responsible for collecting statistics about oracles failures. They proposed to reward oracles for submitting attestations for incorrectness or non-availability. On-chain and off-chain oracles statistics are publicly accessible to users, in real time. As a way to reinforce the validation system, Chainlink implemented a reputation system that allows users themselves to rate the oracles they operated with to fulfill requests. Thus, Chainlink provides global overview to users about existing oracles and their performance. Thus, the choice of oracles becomes more convenient to users based on real objective statistics and witnesses.

4 EXPERIMENTAL TESTBED FOR CHAINLINK

In this section, we describe our evaluation platform for Chainlink. We first describe how we performed the experiments to evaluate the performance of a single Chainlink node. We then proceed to design and implement a decentralized architecture and evaluate its behaviour.

4.1 Motivation

In fact, a Chainlink node is basically a node that runs the Chainlink software on a chosen resource type. As we explained earlier, Chainlink node is splitted into an on-chain part (the oracle smart contract and Ethereum client) and an off-chain part (core and external adapters) and communicates with external data sources that play a principal role in its ecosystem. It is indeed the elementary entity of a decentralized oracle network, built to transfer data between smart contracts and off-chain. Considering the diversity of its components, their deployment methods, their types and configuration parameters options, we decided to evaluate a Chainlink node's behaviour regarding the variation of a selection of its internal parameters and ecosystem characteristics.

4.2 Configuration for Chainlink nodes

Local vs. Cloud-based Chainlink node: The Chainlink software can fluidly run on local machines or on Cloud servers. For a Chainlink node operator, the choice of the resource type is critical as it can presumably affect its qualitative performance in terms of scalability, availability and fault tolerance, and quantitative performance in terms of response time and cost. For this reason, it has a crucial impact on its reputation in the network of Chainlink and then straight influences the node operator earnings. Every Chainlink node calculates its response time, it is the time from recognising the job request on-chain to receiving the transaction receipt and block confirmations after sending the result back on-chain.

For this reason, it is important to lead a comparative study between local Chainlink nodes and cloud-based Chainlink nodes. For quantitative performance measurements, the main decisive metric in a user's perspective and node operator's perspective is the Chainlink node's response time and cost as they allow to evaluate the node performance.

Core vs. External adapters: Another important characteristic that marks a key role in Chainlink is adapters. Adapters are defined as the tool used by nodes to retrieve data from external sources. While core adapters are natively built-in tasks within the Chainlink node, external adapters run separately from the Chainlink native software. However, for in common tasks that can be fulfilled by both core and external adapters, a question is raised regarding the type of adapter to choose. Again, the main metric in terms of quantitative performance that can decide which kind of adapter to choose is response time. In this context, in the Cloud solution, we implemented external adapters through serverless functions.

Serverless Computing is a approach of Cloud Computing, specifically designed for ephemeral, stateless and event-based applications. It relies on a horizontal and on-demand scaling approach. It also assimilates the pay as you go technique of cloud computing. Among the serverless categories are the Function-As-A-Service (or FAAS) [13]. This model allows users to execute their programs that are event-triggered more fluently. In this case, the users servers break down into a set of functions that can be scaled automatically and independently. All the server management and maintenance is the responsibility of the Cloud provider. Besides, serverless computing is economic in terms of cost as the user is only billed for server active state, in other words, no cost is demanded for the idle state of the server which is not the case with traditional Cloud services like IAAS [14]. On the other hand, serverless functions event-driven and server-client characteristics suit the context of oracles as smart contracts can be considered as request-emitters. These requests will create events on the blockchain and will trigger via the Chainlink node the serverless functions.

Chainlink software and hardware parameters: Once the Chainlink node deployment method is investigated and after studying its components and connections, we decided to analyse in the first place its hardware settings in terms of disk type and memory size. It is important for node operators to know the nodes minimum hardware requirements to be able fulfill requests correctly without getting overwhelmed. Second, since the whole Chainlink mechanism is in connection to blockchain, we analyzed its performance when varying Ethereum client providers. Each provider relies on

its own methods to manage its resources balancing and architecture, thus, the Ethereum clients performances can influence the Chainlink node's ones. Another key element in the Chainlink node are initiators. Particularly, we explored Runlog and Cron. Runlog is a typical initiator that watches blockchain log events for a specific JobID, the request must be initiated by the User smart contract. Cron, on the other hand, manages scheduled jobs, in other words, the Chainlink node sends periodically data to a specific smart contract without any initial request needed. The job can be launched autonomously in an interval of seconds, minutes or hours. Finally, extending the scope of the study toward Chainlink ecosystem, we led an analysis about Chainlink nodes attitude toward different data sources. At the same time, it is relevant to observe the distinct data sources performance to confirm our interpretations.

Chainlink centralized architecture: In order to independently evaluate a single Chainlink node in regards to its deployment method, adapters and a set of its software and hardware parameters each time, we implemented a centralized Chainlink system. Certainly, centralization is not our objective since a centralized model is insecure and Chainlink is even based on decentralization. However, we focus in this stage on a single oracle behaviour in the blockchain ecosystem in order to evaluate its performance. The designed architecture is represented by the Figure 4. A user smart contract initiates a request through a blockchain transaction to get a specific data from an external source. The request will be detected by the Chainlink node that will interpret it by following the instructions indicated in the request. If the job calls a core adapter, then the job will be fulfilled internally by the Chainlink node that will directly request data from the external data source. Else if the job calls an external adapter, then the Chainlink node will send it a request to retrieve data from the data source. Once the Chainlink node gets the required data from the external adapter server, it sends it back on-chain to the user smart contract. Depending on the purpose of the experiment, the Chainlink node can be deployed locally or on Cloud services, the adapter can be internal or an external adapter built locally or through a serverless function. Finally, we chose to work with external Web APIs. For experiments in relation to Chainlink node software and hardware parameters, we run our Chainlink node on Cloud services.

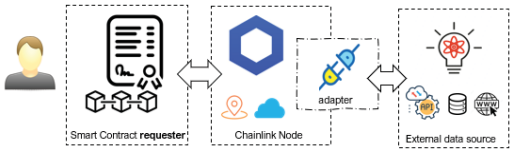

Figure 4: Design of the centralized architecture

4.3 Decentralized network design

Designing and building a centralized architecture was primarily aimed to analyse Chainlink oracle hardware and software setup in order to find out the optimal options to follow as a Chainlink user in general and a node operator in particular. However, a centralized Chainlink system is definitely vulnerable as the overall security is ensured through decentralization, in addition to reputation. We wherefore designed, implemented and evaluated a decentralized Chainlink-based system comprising three Chainlink nodes and two external data sources illustrated by Figure 5. The system is composed of a smart contract that will request data from the Chainlink nodes. Chainlink node 1 and 2 will requests the same data source through the same external adapter while Chainlink node 3 will request a different data source via a different external adapter. Thus, we ensure oracles decentralization as well as data source decentralization.

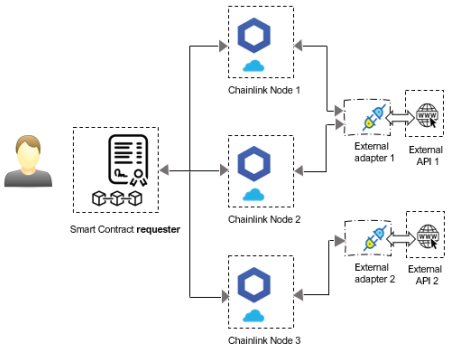

Figure 5: Design of the decentralized architecture

5 RESULTS AND PERFORMANCE ANALYSIS

In this section, we start with describing the setup of our experiments and methodology. Second, we present the results of our experiments. We present and discuss the performance analysis of a Chainlink node on the hardware and software scale. Then we show the results of our implemented decentralized architecture and evaluate its performance.

5.1 Setup and experimental methodology

Chainlink node setup: For both of our local and on cloud nodes, we implemented systems that are very close in terms of hardware characteristics: 8GB of RAM for both local and cloud machines, HDD storage for both solutions and CPU of 2.5Ghz for local machine and 2.7 Ghz for Cloud machine.

In software perspective, The Chainlink team recommends to run Chainlink with Docker as they actively update the docker image from their publicly available repository on Github. Within the Chainlink software, there is a need for a PostgreSQL database server to store information about transactions and global Chainlink execution circumstances. Also, since the Chainlink framework can only be deployed on Ethereum for the moment, an Ethereum client is fundamental. This can be accomplished by running Geth, Parity or a third party like Infura that provides an endpoint for a client Ethereum provider. As a result, we configured our local and cloud machines as follow: for both local and cloud machines we exploited Infura for Ethereum client. we run Chainlink via latest docker image, and we used PostgreSQL 12. For the cloud-based node, we chose to implement it on Google Cloud Platform.

Adapters setup: While Core adapters are built-in services, external adapters are self-developed tools. They are are as simple as a server

listening to clients and fulfilling requests. It can be implemented in Python or NodeJS. For the local node we ran the EA server locally on Visual studio Code, while for the cloud-based node we launched the EA server via serverless Google Functions.

For our experiments with both local and cloud-based nodes, we ran NodeJS servers that return the ETH price in USD. In this context, we utilized the cryptocompare API[2]. Figure 6 shows the architecture implemented on Google Cloud services. We utilized Google Compute Engine to run the virtual machine, a Google PostgreSQL server for the database and Google Functions to run the external adapters. For the local system, we run the Chainlink software and PostgreSQL on the same machine, and we developed our local external adapters using NodeJS. The request of the smart contract specifies the job to consider by the Chainlink node. In the first workflow, a Chainlink node has to use its built-in adapter to get the data. While in the second workflow, the node has to send a request to an external adapter in order to retrieve the data.

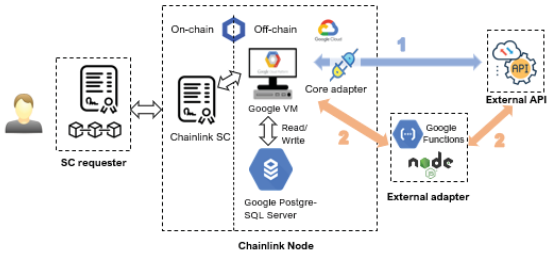

Figure 6: Centralized Chainlink system on Google

Decentralized system setup: For our decentralized architecture, we built our nodes on Google machines, using Compute Engine service, configured as follow: 3.75GB memory, HDD disk, Runlog initiator, Infura Ethereum provider and external adapters deployed on serverless functions using Google Functions. Both data sources are IP tracking APIs that for a specified IP will send a set of information like country, region, city, latitude, longitude, currency, etc. We developed the adapters using NodeJS and programmed them to retrieve latitude and longitude from the API response. The information is then forwarded back to the smart contract. The APIs used are IPregistry[3] and IPwhois[4].

To evaluate the overall system, we recorded response time for each Chainlink node, Ethereum transaction fees in ETH, Chainlink transaction fees in LINK and Cloud fees for a total of 500 requests divided into five intervals. For all our experiments we were based on Ethereum blockchain platform, as Chainlink runs currently on Ethereum only. Also, all our systems are deployed on Kovan Ethereum network.

Methodology: To analyze the performance of Chainlink nodes, we programmed a smart contract to launch requests to Chainlink nodes in order to retrieve data. The Chainlink GUI provides the response time for each request. For adapters response time, we were able to retrieve it from Chainlink command line logs. In order to properly conduct our experiments, we proceeded in the following methodology:

(1) For each Chainlink node parameter analysis, we implement multiple centralized systems in parallel, all the internal and external configurations are perfectly similar except for the parameter we intend to investigate. In order to survey the systems performance, we send requests consecutively (once the request of the first system is fulfilled, we immediately launch the request for the second system). In that case, the evaluation will not depend on the blockchain circumstances (Example: Network congestion, ETH gas price), rather it will effectively measure the nodes performance.
(2) we conducted the experiments on different time intervals. Time interval in this context is the temporal difference between a number of requests. For example, we send requests, after 2 hours, we send requests again and after 14 hours we re-send requests, for both solutions. In such a way, the evaluation will not depend on the time factor.
(3) In every time interval, we vary the number of requests. For example , in time interval T1 we send 15 consecutive requests, in T2 we send 25 requests and in T3 we send 10 requests.
(4) To enhance the time Independence, for each interval we vary the time between requests for the same solution. For example, between request 1 and 2, we wait 50 seconds, between request 2 and 3 we wait 5 minutes and between requests 3 and 4 we wait 70 seconds.

5.2 Evaluation of Chainlink oracles

Impact of Chainlink Node deployment method: Deciding the Chainlink node deployment method is a critical issue for Chainlink node operators as it determines the performance level and has a considerable impact on the operator reputation in the Chainlink network. Figures 7 and 8 show respectively the cumulative distribution function (CDF) for the response time of local and cloud-based Chainlink nodes using core adapter and external adapter. For this purpose we launched 800 requests for each system.

We can notice that for both experiments, the local node is faster to fulfill requests than the Cloud-based one. The response times for both Chainlink machine types are between 13 and 37 seconds. For 55% of the requests, local Chainlink node is 2.5 seconds faster using core adapter and 3 seconds faster using an external adapter. In the first system (represented by Figures 7), a local machine takes an average of 21.97 seconds to respond to a request while a cloud-based machine takes 22.76 seconds. Also, in 64.08% of requests, response time is less than 22s in local machine against 52.59% for cloud machine. Regarding the solutions cost, obviously local machine costs no fees except for electricity while the cloud-based Chainlink node costs 126.69 USD working 24/7. As for a beginner node operator, we recommend starting with a local Chainlink machine as it is a bit faster, for free and since the income is 100% earned, it is considered as a good motivating beginning.

Once the operator fits deeply in the Chainlink network and gets involved with multiple Chainlink users and applications, we recommend switching to cloud-based machines, although local machines are faster. In fact, every Chainlink node reputation is publicly available and shows the response ratio. A response ratio

[2]Cryptocompare: https://min-api.cryptocompare.com
[3]https://ipwhois.io/
[4]https://ipregistry.co/

reflects the percentage of the node availability to fulfill requests and a low one gives an inappropriate idea to users. Moreover, depending on the application nature, many users would rely foremost on response ratio in order to decide which Chainlink node operator to opt for. Therefore, high availability and scalability must be key characteristics in Chainlink nodes. In this context, Cloud providers offer these services, that are hard to attain in local systems, in addition to maintenance, easier upgrades and flexible hardware and software adjustments.

Impact of adapter type: We mentioned earlier that for a good set of applications, both of core adapters and external adapters can be applied. Thus, we monitored for local and cloud-based systems the response time of core adapter and external adapter. Figure 9 presents the CDF of response time for both adapters in a local system and Figure 10 corresponds to the cloud-based one. The response times were recorded from the experiments presented in the previous paragraph. As for their performance, we can notice that in the local system, the response time for 87% of the requests are nearly analogous and are between 0.475s and 1s. The difference is more explicit in the cloud-based system where core adapter on average is 0.3s faster to respond to requests. 84.84% of core adapter response time are less than 0.63s while for 85.05% of requests, response time is between 0.67s and 0.98s. Thus, core adapters are slightly faster to respond, they are recommended in case of simple requests. However, external adapter becomes more interesting for more complex APIs as it is completely customizable and interoperable. In particular, for Google Functions, 2500 invocations per month for one light serverless function cost 0.84 USD.

Impact of hardware setup: On the hardware scale, first, we analysed Chainlink nodes performance based on the memory parameter. We tested three possibilities via three cloud machines, all parameters are identical except for memory: 0.6G, 1.7G and 3.75G. For an example of 300 requests, we recorded the response time. The results are presented as CDF are shown in Figure 11. Here the performance is practically uniform. 3.75G setup is at the forefront of response time with a slight difference time. It is still important to mention that for 0.6G setup, the machine is more likely to respond after more than 30s, for 0.2% against 0.006% in other setups.

On the other hand, since the Chainlink node actively writes blockchain information into the PostgreSQL database, we compared its performance based on its disk type, HDD and SSD. The results, shown in Figure 12, demonstrate that response time is not affected by this parameter.

In this context, 3.75G memory and HDD disk cloud machine is adequate. besides, in our hardware setup, we run our machines with 2.25Ghz CPU, where during all the experiments, only 3% of the CPU capacity was utilized.

Impact of initiator type: Besides to hardware setup, there is a set of Chainlink software parameters that an operator has to fix. In this paragraph we present a comparison between two Chainlink initiators performance: Runlog and Cron. Figure 13 evaluates Chainlink nodes response time for each solution. In fact, results demonstrate that Cron has a huge favorable influence. Response times are between 13s and 17s. 89% of them are equal to 13s. For Runlog initiator, response time is between 20s and 25s. Actually, the Chainlink machine gains much time when periodically and automatically fulfilling its task. In this way, it does not have to watch over the blockchain logs and process the task. In case of frequent requests and recurring tasks, Cron initiator is suitable.

Impact of Ethereum provider: On the blockchain Scale, we evaluated the response time depending on the Ethereum provider. We utilized Infura, Quicknode and Alchemy in 300 requests experiment. These services provide Ethereum core infrastructure, helping blockchain ecosystem scale and empower a lot of blockchain users and decentralized applications. Figure 16 shows the impact of varying the Ethereum providers. As we can notice, the solutions performances are harmonious. For response time less than 22s, the Quicknode system is at the lead, having 88.23%, Alchemy has 84.03%, and Infura has 78.63%. Moreover, Quicknode has the least of requests lasting for 24s or more, 11.76% compared to 16.96% for Alchemy and 14.52% for Infura. The difference in performance exists but is not profound. Regarding Ethereum providers minimum pricing, Infura offers managing 100K/day Ethereum requests for free, Quicknode offers 300K/day requests for 9 USD and Alchemy offers 100,000K compute units per month for free. As for quality-price ratio, Quicknode is a good choice.

Impact of data source type: Since Chainlink oracles are build mainly to retrieve data from the off-chain world, they have a vital and permanent bond with external data sources. In this context, oracles requests fulfillment highly depends on the response rate of data sources. We therefore evaluated the Chainlink node response time based on two different data sources, cryptocompare API and a geolocalisation API, IPregistry. For a set of 300 requests, Figure 14 that presents this variation, clearly proves a noteworthy dissimilarity in oracle performance. For 73% of requests, there is a 4 seconds gap in response time. For IPregistry API, 89% of requests are between 20s and 21s while 85% of requests are between 24s and 25s in the Cryptocompare solution.

This discard is affirmed by Figure 15 where it illustrates the variation of response time of each API. IPregistry responses in 95% of requests within 0.1s up to 0.5s, while Cryptocompare answers for 60% of requests in the range of 3.5s and 6s.

Results for decentralized architecture: For each interval of 100 requests, we recorded and calculated the average response time for each Chainlink machine, the Ethereum cost, Cloud cost and LINK cost. The measurements are presented in Table 1.

As we analysed in our previous paragraphs, the response time and cost depend on several internal and external parameters. Evidently, the cost of such architecture and in-bound mechanism is not static as it permanently depends on the blockchain network. As the Ether price is actively variable, 100 requests can cost 0.1129 Ether as it can cost up to 0.4764 Ether and more. In the Mainnet Ethereum scope and according to Google Finance Ether price statistics from September 2020[5], the system's Ether price for 100 requests costing 0.1129 Ether can go from 38.39 USD up to 471.80 USD. Regarding LINK cost, it is actually fixed by the node operator, that is, for each job it offers, the node operator estimates and decides the cost in LINK for each task he fulfills. For instance, we configured our nodes in this system to fulfill jobs for 0.1 LINK/request.

For each request interval, the average of nodes response time is nearly equal. In other words, in the same hardware, software and blockchain circumstances, Chainlink nodes behave in a harmonious

[5]https://www.google.com/finance/quote/ETH-USD

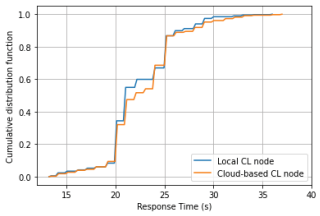

Figure 7: Core adapter

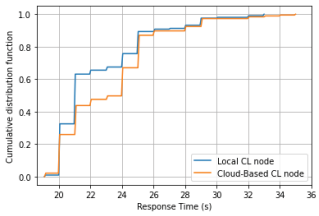

Figure 8: External adapter

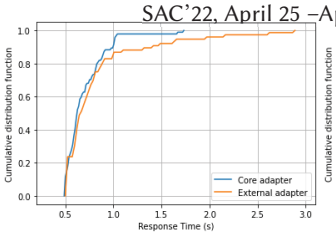

Figure 9: Local node

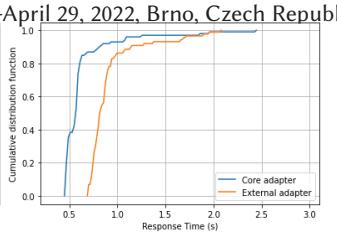

Figure 10: Cloud node

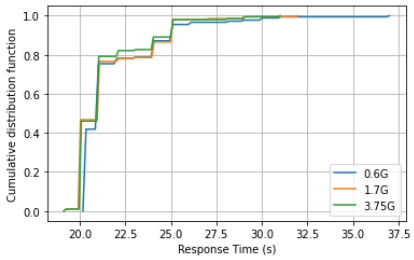

Figure 11: Memory size

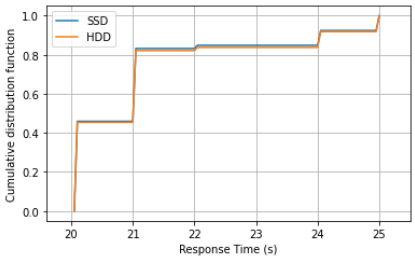

Figure 12: Disk type

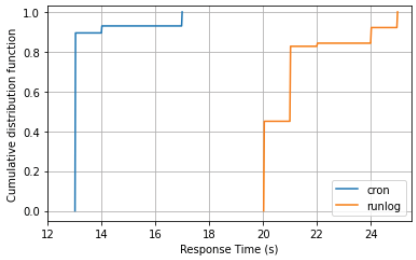

Figure 13: Initiator type

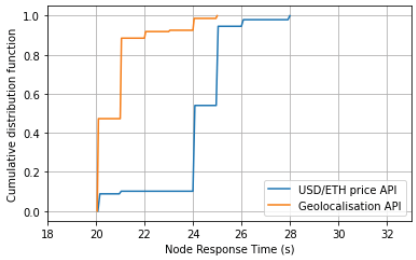

Figure 14: Node response time

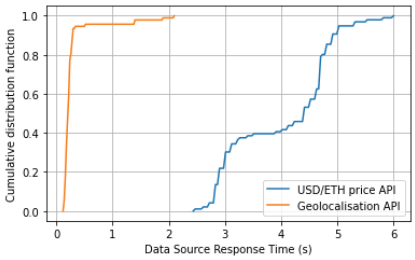

Figure 15: Data source response time

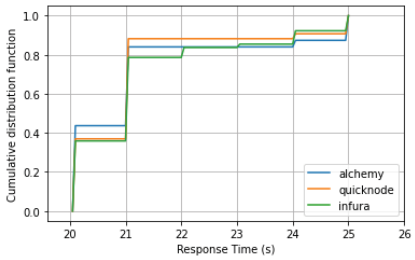

Figure 16: Ethereum provider type

Table 1: Decentralized Chainlink network performance

Requests	Avg. resp. time, Node 1 (s)	Avg. resp. time, Node 2 (s)	Avg. resp. time, Node 3 (s)	Ethereum cost (ETH)	Cloud cost (USD)	Chainlink cost (LINK)
100	21.91	22.08	22.24	0.1263	11.05	30
100	22.07	21.66	21.43	0.2181	12.21	30
100	21.10	21.14	21.36	0.4764	10.85	30
100	21.25	21.45	21.59	0.1129	12.32	30
100	20.62	20.69	20.46	0.1830	11.52	30

similar manner. The variance starts from 0.04s and reaches 0.64s. The overall response time average for this system to fulfill a smart contract demand is 21.59 second/request. As we proved earlier, a number of parameters such as data source, Chainlink machine type, Ethereum provider and initiator type can have uneven effects on oracles performance. Thus, these parameters can influence nodes response time and cost. Also, it can affect the node's availability, in case of a local machine.

In order to lead deeper investigations regarding the network performance we statistically illustrated the response times in the following pie chart. We mixed all nodes response time in one data entity.

As we can notice in Figure 17, 75% of requests are fulfilled within 20s or 21s. Request times are between 13s and 45s. However, the number of requests that are at the extremity (inferior to 20s or superior to 31s) present only 4%. This reflects a harmonious behavior of Chainlink oracles all along the period of experiments for the decentralized system (2 months). We note that 93% of requests are fulfilled in 25s or less. Unfortunately, there is no previous research work that experimentally evaluates Chainlink and other oracle

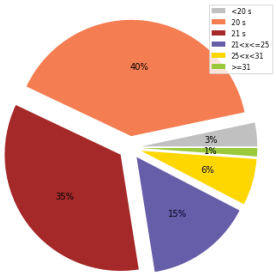

Figure 17: Pie chart for Chainlink oracles response time

frameworks like Town Crier or Provable, we therefore cannot compare Chainlink real performance in terms of response times to other oracle solutions. However, our benchmarking results prove that Chainlink performance is relevant. Evidently, the response time of the system as a whole depends on the aggregation mechanism adopted to gather and generate the final results. For instance, if a system of 10 oracles will generate the final result after receiving only 3 responses, the system's response time will be likely faster than waiting the 10 answers. On the other hand, security could be less solid, and cost would raise issues, this belongs to our future work, continuing to experimentally explore Chainlink.

6 CONCLUSION

In this research work, our goal is to cover the missing experimental study of Chainlink, we therefore analyzed the Chainlink network global performance and investigated a single Chainlink node behavior in regards to the variation of a set of decisive parameters and configurations. This builds the first research evaluation of Chainlink based on devoted implemented systems.

First, we investigated the Chainlink node deployment method. Results show that a local node is faster than a Cloud one, on the other hand, the Cloud machine is obviously more expensive. Regarding adapters, the performance shows that core adapters are slightly faster, however, external adapters are more likely to fit into any application, which tackles the core adapters limit. As node operators get deeper basis in the Chainlink community, we recommend relying on external adapters and Cloud machines to guarantee high availability, scalability and more flexible maintenance and adjustments.

With respect to hardware setup, we can estimate that a machine of 3.75G memory, HDD disk and 2.25Ghz is widely sufficient to successfully and properly run a Chainlink node. Our evaluations also prove that Cron initiator is much more efficient than Runlog. Cron is very favorable for smart contracts that periodically request external data. The data source also has a relevant impact on the Chainlink node performance. Also, we tested different Ethereum Client providers, the results were almost identical, the choice can eventually depend on the best package offered, in this context, we recommend using Quicknode.

In regards to the decentralized Chainlink network we implemented, it provides to Chainlink users a real world Chainlink-based system performance in terms of response time and cost. the average response time is in the range of 21-22s. As for the system's cost, it highly varies depending on the ETH price, that is variable and currently expensive.

As for our future work, we plan to implement an optimization mechanism, in the decentralization stage, to ensure balance between the system security, latency and cost regarding the number of nodes and data sources to activate. This belongs actually to the aggregation technique that has to be implemented.

REFERENCES

[1] Buterin, V. A next generation smart contract & decentralized application platform (2013) whitepaper. Ethereum Foundation.
[2] H. Al Breiki, L. Al Qassem, K. Salah, M. Habib Ur Rehman and D. Sevtinovic, "Decentralized Access Control for IoT Data Using Blockchain and Trusted Oracles," 2019 IEEE International Conference on Industrial Internet (ICII), 2019, pp. 248-257, doi: 10.1109/ICII.2019.00051.
[3] Abdeljalil, B. (2020). A Study of Blockchain oracles. Retrieved from: https://arxiv.org/abs/2004.07140.
[4] Hamda, A.-B., Muhammad, H. U. R., Khaled, S. & Davor, S. (2017). TrustworthyBlockchain oracles: Review, Comparison, and Open Research Challenges.IEEEAccess. doi: 10.1109/ACCESS.2020.2992698
[5] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira and A. Kastania, "Astraea: A Decentralized Blockchain Oracle," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018, pp. 1145-1152, doi: 10.1109/Cybermatics_2018.2018.00207.
[6] H. Al-Breiki, M. H. U. Rehman, K. Salah and D. Svetinovic, "Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges," in IEEE Access, vol. 8, pp. 85675-85685, 2020, doi: 10.1109/ACCESS.2020.2992698.
[7] Petar Kochovski, Sandi Gec, Vlado Stankovski, Marko Bajec, Pavel D. Drobintsev, Trust management in a blockchain based fog computing platform with trustless smart oracles,Future Generation Computer Systems,Volume 101,2019, https://doi.org/10.1016/j.future.2019.07.030.
[8] Weisheng Lu, Xiao Li, Fan Xue, Rui Zhao, Liupengfei Wu, Anthony G.O. Yeh, Exploring smart construction objects as blockchain oracles in construction supply chain management, Automation in Construction, Volume 129,2021,103816, https://doi.org/10.1016/j.autcon.2021.103816.
[9] Caldarelli, G. Understanding the Blockchain Oracle Problem: A Call for Action. Information 2020, 11, 509. https://doi.org/10.3390/info11110509
[10] Basile D., Goretti V., Di Ciccio C., Kirrane S. (2021) Enhancing Blockchain-Based Processes with Decentralized Oracles. In: González Enríquez J., Debois S., Fettke P., Plebani P., van de Weerd I., Weber I. (eds) Business Process Management: Blockchain and Robotic Process Automation Forum. BPM 2021. Lecture Notes in Business Information Processing, vol 428. Springer, Cham. https://doi.org/10.1007/978-3-030-85867-4_8
[11] Zhang, F., Cecchetti1. E., Croman1, K., Juels2, A., Shi, S. (2021). Town Crier: An Authenticated Data Feed for Smart Contracts, retrieved from: https://towncrier.org/files/2016/168.pdf.
[12] Adán, S. d. P. A., Daniele, L. & Luis, I. C. Witnet: A Decentralized oracle Network Protocol (2017) Whitepaper.
[13] Steve, E., Ari, J. & Sergey, N. ChainLink A Decentralized oracle Network (2017) Whitepaper.
[14] van Eyk, E., Iosup, A., Seif, S. & Thömmes, M. (2017). The SPEC Cloud Group's ResearchVision on FaaS and Serverless Architectures.Proceedings of the 2nd InternationalWorkshop on Serverless Computing, pp. 1–4. doi: 10.1145/3154847.3154848.
[15] Eric, J., Johann, S.-S., Vikram, S., Chia-Che, T., Anurag, K., Qifan, P., Vaishaal, S., Joao,C., Karl, K., Neeraja, Y., Joseph, E. G., Raluca, A. P., Ion, S. & David, A. P. (2019).Cloud Programming Simplified: A Berkeley View on Serverless Computing. Re-trieved from: arXiv:1902.03383

ANNEXE II

EXEMPLE D'UN CONTRAT INTELLIGENT DANS LE CONTEXTE DE CHAINLINK

La figure suivante illustre un extrait d'un contrat intelligent écrit en Solidity qui permet de lancer une requête à un nœud Chainlink pour récupérer des données externes. l'attribut ORACLE_-ADDRESS définit l'adresse du nœud Chainlink et JOBID spécifie l'adresse du Job à exécuter par le nœud. Pour lancer une requête, l'utilisateur doit faire appel selon une transaction à la fonction requestEthereumPrice. Une fois la requête est satisfaite et la réponse est en chaîne, le retour du nœud Chainlink est stocké dans la variable currentPrice par l'intermédiaire de la fonction fulfill.

```
pragma solidity ^0.6.0;

import "github.com/smartcontractkit/chainlink/evm-contracts/src/v0.6/ChainlinkClient.sol";


contract ChainlinkExample is ChainlinkClient {
  uint256 public currentPrice;
  address public owner;
  address ORACLE_ADDRESS = 0x02A89CE429f285c691ba975c4eeBBa8E18755039;
  string constant JOBID = "9560e51491704cd8b2ad8e50185534d3";
  uint256 constant private ORACLE_PAYMENT = 100000000000000000;

  constructor() public {
    setPublicChainlinkToken();
    owner = msg.sender;
  }

  function requestEthereumPrice()
    public
    onlyOwner
  {
    Chainlink.Request memory req = buildChainlinkRequest(stringToBytes32(JOBID), address(this), this.fulfill.selector);
    sendChainlinkRequestTo(ORACLE_ADDRESS, req, ORACLE_PAYMENT);
  }

  function fulfill(bytes32 _requestId, uint256 _price)
    public
    recordChainlinkFulfillment(_requestId)
  {
    currentPrice = _price;
  }

```

Figure-A II-1 Extrait d'un contrat intelligent écrit en Solidity

ANNEXE III

EXEMPLE D'UN JOB CHAINLINK

Un job est une description du travail qu'un nœud Chainlink doit effectuer, exprimé en fonction de l'initiateur et adaptateurs (tasks). La figure suivante illustre un Job écrit sous format JSON, relatif à un nœud Chainlink. Dans l'exemple, l'initiateur à utiliser est Runlog, en spécifiant, en option, l'adresse du contrat intelligent cible. Les adaptateurs utilisés sont : un adaptateur externe nommé cryptocompare, l'adaptateur ethuint256 qui formate son entrée en un entier puis le convertit au format Solidity uint256 et finalement l'adaptateur ethtx qui prend l'entrée donnée et la place dans le champ de données de la transaction, il signe ensuite une transaction Ethereum et la diffuse sur le réseau.

```
{
    "initiators": [
        {
            "type": "runlog",
            "params": {
                "address": "0xeb171b73159d4f0244e2293c3b439e01a6fbd6da"
            }
        }
    ],
    "tasks": [
        {
            "type": "cryptocompare",
            "confirmations": null,
            "params": {
                "to": "USD",
                "from": "ETH"
            }
        },
        {
            "type": "ethuint256",
            "confirmations": null,
            "params": {}
        },
        {
            "type": "ethtx",
            "confirmations": null,
            "params": {}
        }
    ],
    "startAt": null,
    "endAt": null
}
```

Figure-A III-1 Un exemple de Job dans Chainlink

BIBLIOGRAPHIE

Abdeljalil, B. (2020). A Study of Blockchain Oracles. Repéré à https://arxiv.org/abs/2004.07140.

Adler, J., Berryhill, R., Veneris, A., Poulos, Z., Veira, N. & Kastania, A. (2018). Astraea : A Decentralized Blockchain Oracle. *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1145-1152. doi : 10.1109/Cybermatics_2018.2018.00207.

Adán, S. d. P. A., Daniele, L. & Luis, I. C. (2017). Witnet : A Decentralized Oracle Network Protocol.

Bouichou, A., Mezroui, S. & Oualkadi, A. E. (2020). An overview of Ethereum and Solidity vulnerabilities. Dans *2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)* (vol. 25, pp. 1-7). doi : 10.1109/ISAECT50560.2020.9523638.

Breiki, H. A., Qassem, L. A., Salah, K., Rehman, M. H. U. & Sevtinovic, D. (2019). Decentralized Access Control for IoT Data Using Blockchain and Trusted Oracles. *2019 IEEE International Conference on Industrial Internet (ICII)*, 248-257. doi : 10.1109/ICII.2019.00051.

Buterin, V. (2013). A next generation smart contract & decentralized application platform.

Caldarelli, G. Understanding the Blockchain Oracle Problem : A Call for Action. doi : https://doi.org/10.3390/info11110509.

D., B., V., G., C., D. C. & S., K. (2021). Enhancing Blockchain-Based Processes with Decentralized Oracles. *Internet(eds) Business Process Management : Blockchain and Robotic Process Automation Forum. BPM 2021. Lecture Notes in Business Information Processing, vol 428. Springer, Cham.* doi : https://doi.org/10.1007/978-3-030-85867-4_8.

Eric, J., Johann, S.-S., Vikram, S., Chia-Che, T., Anurag, K., Qifan, P., Vaishaal, S., Joao, C., Karl, K., Neeraja, Y., Joseph, E. G., Raluca, A. P., Ion, S. & David, A. P. (2019). Cloud Programming Simplified : A Berkeley View on Serverless Computing. Repéré à arXiv: 1902.03383.

Fan, Z., Ethan, C. & Kyle, C. (2016). Town Crier : An Authenticated Data Feed for Smart Contracts. Dans *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 270–282). Association for Computing Machinery. doi : https://doi.org/10.1145/2976749.2978326.

Giessmann, A. & Stanoevska-Slabeva, K. (2012). Business Models of Platform as a Service (PaaS) Providers : Current State and Future Directions. *JITTA : Journal of Information Technology Theory and Application*, 13(4), 31-54. Repéré à https://www.proquest.com/scholarly-journals/business-models-platform-as-service-paas/docview/1430425009/se-2?accountid=27231.

Gustavo, A. O., Ahmed, E. H. & Zhen, M. J. J. (2020). An exploratory study of smart contracts in the Ethereum blockchain platform. *Empirical Software Engineering*, 25. doi : https://doi.org/10.1007/s10664-019-09796-5.

Hamda, A.-B., Muhammad, H. U. R., Khaled, S. & Davor, S. (2017). Trustworthy Blockchain Oracles : Review, Comparison, and Open Research Challenges. *IEEE Access*. doi : 10.1109/ACCESS.2020.2992698.

Ilham, A. Q., Manar, A. T. & Ali, B. N. (2018). Performance Analysis of Hyperledger Fabric Platforms. 2018, 1-14. doi : https://doi.org/10.1155/2018/3976093.

Insights, I. M. D. . (2021). Serverless in the enterprise, 2021 : Building the next generation of efficient, flexible, cost-effective cloud native applications. Repéré à https://www.ibm.com/downloads/cas/ZJLWQOAQ.

Kochovski, P., Gec, S., Stankovski, V., Bajec, M. & Drobintsev, P. D. (2019). Trust management in a blockchain based fog computing platform with trustless smart oracles. *Future Generation Computer Systems*, 101. doi : https://doi.org/10.1016/j.future.2019.07.030.

Lu, W., Li, X., Xue, F., Zhao, R., Wu, L. & Yeh, A. G. Exploring smart construction objects as blockchain oracles in construction supply chain management. doi : https://doi.org/10.1016/j.autcon.2021.103816.

Lukman, A. A., ORCID, J. A., ORCID, E. A. A. & Loveth, K. (2019). Crypto Hash Algorithm-Based Blockchain Technology for Managing Decentralized Ledger Database in Oil and Gas Industry. doi : https://doi.org/10.3390/j2030021.

Michael, N., Peter, G., Oliver, H. & Schiereck, D. (2017). Blockchain. doi : 10.1007/s12599-017-0467-3.

Mudabbir, K. & Weidong, S. (2021). Demystifying Pythia : A Survey of ChainLink Oracles Usage on Ethereum. Repéré à https://arxiv.org/pdf/2101.06781.pdf.

Nakamoto, S. (2008). Bitcoin : A Peer-to-Peer Electronic Cash System.

NIST. (2011). The NIST Definition of Cloud Computing. Repéré à https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

Prodan, R. & Ostermann, S. (2009). A survey and taxonomy of infrastructure as a service and web hosting cloud providers. Dans *2009 10th IEEE/ACM International Conference on Grid Computing* (pp. 17-25). doi : 10.1109/GRID.2009.5353074.

Shuai, W., Yong, Y., Xiao, W., Juanjuan, L., Rui, Q. & Fei-Yue, W. (2018). An Overview of Smart Contract : Architecture, Applications, and Future Trends. Dans *2018 IEEE Intelligent Vehicles Symposium (IV)* (pp. 108-113). doi : 10.1109/IVS.2018.8500488.

Sin, K. L., Xiwei, X., Mark, S. & Lina, Y. (2020). Reliability analysis for blockchain oracles. *Computers & Electrical Engineering*, 83, 106582. doi : https://doi.org/10.1016/j.compeleceng.2020.106582.

Steve, E., Ari, J. & Sergey, N. (2017). ChainLink A Decentralized Oracle Network.

Tareq, A., Arman, S., Saman, S., Daniels, J. & Ben, A. (2017). Blockchain Technology Innovation. *IEEE Technology & Engineering Management Conference*.

Van Eyk, E., Iosup, A., Seif, S. & Thömmes, M. (2017). The SPEC Cloud Group's Research Vision on FaaS and Serverless Architectures. *Proceedings of the 2nd International Workshop on Serverless Computing*, pp. 1–4. doi : 10.1145/3154847.3154848.

Xiaoqiong, X., Gang, S., Long, L., Huilong, C., Hongfang, Y. & Athanasios, V. (2021). Latency performance modeling and analysis for hyperledger fabric blockchain network. 58, 102436. doi : https://doi.org/10.1016/j.ipm.2020.102436.