Elrond

Une Blockchain publique hautement scalable grâce au partitionnement horizontal (sharding) d'états adaptatif et au consensus à preuve d'enjeu sécurisée (Secure Proof of Stake - SPoS)

[Livre blanc technique - Edition 2 - révision 1]

mises à jour des chapitres: [5 - 13] 19 Juin 2019 – L’équipe Elrond https://www.elrond.com/

Résumé - L'avènement de la Blockchain publique sécurisée avec Bitcoin puis Ethereum, a suscité un intérêt notable et l'afflux de capitaux, fournissant les prémisses d'une vague mondiale d'innovation ouverte à tous. Malgré de grandes promesses, la création d'une Blockchain décentralisée, sécurisée et évolutive s'est révélée être une tâche ardue.

Ce document est consacré à Elrond, une nouvelle architecture qui transcende l'état de l'art en introduisant un véritable mécanisme de partitionnement d'états (State Sharding) clé de voute d'une scalabilité opérationnelle, éliminant les gaspillages d'énergie et de calcul tout en garantissant une équité distribuée grâce à un consensus à preuve d'enjeu sécurisée (SPoS). En mettant l'accent sur la sécurité, le réseau d'Elrond est construit pour résister aux problèmes de sécurité connus comme l'attaque Sybil, l'attaque "Rien à perdre" et d'autres. Dans un écosystème qui prône l'interconnectivité, notre solution pour implémenter les contrats intelligents (Smart Contracts) offre un moteur compatible avec EVM afin d'inscrire l'interopérabilité dans son ADN.

Les simulations préliminaires et les résultats du testnet montrent qu'Elrond dépasse le débit moyen de Visa en l'améliorant d'un facteur supérieur à 3, ou, d'un facteur 1000 par rapport aux approches viables existantes, tout en réduisant considérablement les coûts d'amorçage et de stockage pour assurer la durabilité à long terme.

I Introduction

1 Aspects généraux

Les plates-formes de crypto-monnaie et de contrats intelligents (Smart Contracts) telles que Bitcoin et Ethereum ont suscité un intérêt considérable et sont devenues des solutions prometteuses pour les paiements électroniques, les applications décentralisées et les réserves numériques potentielles de valeur. Toutefois, en comparaison des indicateurs clés de leurs homologues centralisés[1], l'état actuel des choses suggère que les versions actuelles des Blockchains publiques présentent de sérieuses limitations, notamment en ce qui concerne la scalabilité, ce qui fait obstacle à leur adoption par le grand public et lui en retarde son utilisation. En fait, il s'est avéré extrêmement difficile de gérer les limites techniques actuelles imposées par les compromis du paradigme du trilemme des Blockchains [2]. Plusieurs solutions ont été proposées, mais peu d'entre elles ont donné des résultats significatifs et viables. Ainsi, pour résoudre le problème de la scalabilité, il a fallu repenser complètement les infrastructures de la Blockchain publique.

2 Définir les défis

Plusieurs défis doivent être correctement relevés dans le processus de création d'une solution innovante de blockchain publique conçue pour une mise à l'échelle:

• Décentralisation complète - Élimination du besoin de tout tiers de confiance, supprimant ainsi tout point de défaillance unique;

Sécurité robuste – Permettant des transactions sécurisées et empêchant toute attaque basée sur des vecteurs connus;

Haute scalabilité - Permettre au réseau d'atteindre une performance en transactions par seconde (TPS) au moins égale à son homologue centralisé;

Efficacité - Exécution de tous les services réseau avec des exigences énergétiques et informatiques minimales;

• Amorçage et amélioration du stockage - Assurant un coût compétitif pour le stockage et la synchronisation des données;

• Interopérabilité entre chaînes - renforcée nativement à la conception, permettant une communication illimitée avec les services externes. À partir des défis ci-dessus, nous avons créé Elrond en repensant complètement l'infrastructure de Blockchain publique, spécialement conçue pour être sécurisée, efficace et interopérable. La principale contribution d'Elrond repose sur 2 piliers fondamentaux de construction:

1) Un véritable mécanisme de partitionnement horizontal d'états (State Sharding) : partitionner efficacement la Blockchain et l'état des comptes en plusieurs fragments, gérés en parallèle par différents validateurs participants;

2) Un mécanisme de consensus sécurisé à preuve d'enjeu : une variante améliorée de la preuve d'enjeu (PoS) qui garantit la sécurité à long terme et l'équité distribuée, tout en éliminant le besoin d'algorithmes PoW énergivores.

3 Partitionnement d'état adaptatif

Elrond propose un mécanisme de partage adaptatif dynamique qui permet le calcul et la réorganisation des fragments en fonction du besoin et du nombre de nœuds réseau actifs. La réaffectation des nœuds dans les fragments au début de chaque époque est progressive et non déterministe, n'induisant aucune pénalité temporaire sur la note de performance. Le partage d'état adaptatif s'accompagne de défis supplémentaires par rapport au modèle statique. L'un des points clés réside dans la façon dont le partitionnement et la fusion des fragments sont effectués pour éviter les pénalités de latence globales introduites par les besoins de synchronisation / communication lorsque le nombre de fragments change. La latence, dans ce cas, est la surcharge de communication requise par les nœuds, afin de récupérer le nouvel état, une fois que leur affectation d'espace d'adressage de fragment a été modifiée.

Elrond propose une solution au problème ci-dessous, mais d'abord quelques notions doivent être définies : utilisateurs et nœuds. Les utilisateurs sont des acteurs externes et peuvent être identifiés par une adresse de compte unique ; les nœuds sont des ordinateurs/appareils du réseau Elrond qui exécutent notre protocole. Des notions telles que les utilisateurs, les nœuds et les adresses seront décrites plus en détail dans le chapitre II.1 - Entités.

Elrond résout ce défi en :

1) Divisant l'espace d'adressage des comptes dans les fragments, en utilisant un arbre binaire qui peut être construit avec la seule exigence de connaissance exact du nombre de fragments à une époque donnée. Cette méthode permet de réduire la latence accumulée et d'améliorer l'efficacité du réseau de deux manières. Premièrement, grâce au modèle conçu, la division de l'espace d'adressage des comptes est prédéterminée par la hiérarchie. Ainsi, il n'y a pas de surcharge de division, ce qui signifie qu'un fragment se divise en deux. Chacun d'eux ne conservant que la moitié de l'espace d'adressage précédent en plus de l'état associé. Ensuite, la latence est réduite grâce au mécanisme de redondance d'état, car la fusion est préparée en conservant l'état dans les nœuds de meme parent.

2) Introduisant une technique d'équilibrage des nœuds dans chaque fragment, afin d'obtenir un équilibre global de l'architecture. Cette technique assure une charge de travail équilibrée et une récompense pour chaque nœud du réseau.

3) Concevant un mécanisme intégré pour l'acheminement automatique des transactions dans les fragments correspondants, ce qui réduit considérablement la latence. L'algorithme de routage est décrit au chapitre IV.4 - Approche des Shards d'Elrond.

4) Afin d'obtenir des améliorations considérables tout en ne pénalisant pas l'initialisation au démarrage et le stockage, Elrond utilise un mécanisme d'élagage des fragments. Cela assure la durabilité de notre architecture, même avec un débit de dizaines de milliers de transactions par seconde (TPS).

4 Preuve d'enjeu sécurisée (SPoS ou Secure Proof of Stake)

Nous introduisons un mécanisme de consensus "Preuve sûre de l'enjeu", qui développe l'idée d'Algorand [3] d'un mécanisme de sélection aléatoire, en se différenciant par les aspects suivants :

1) Elrond introduit une amélioration qui réduit la latence en permettant à chaque nœud des fragments de déterminer les membres du groupe de consensus (proposants de bloc et valideurs) au début d'un tour. Cela est possible parce que le facteur de randomisation r est stocké dans chaque bloc et est créé par le proposant de bloc en utilisant une signature BLS [4] sur le r précédent.

2) Le proposant du bloc est le valideur dans le groupe de consensus dont le hachage de la clé publique et du facteur de randomisation est le plus petit. Contrairement à l'approche d'Algorand [3], où la sélection du comité aléatoire peut prendre jusqu'à 12 secondes, avec Elrond, le temps nécessaire à la sélection aléatoire du groupe de consensus est considérablement réduit (estimé à moins de 100 ms), hors latence du réseau. En effet, il n'y a pas d'exigence de communication pour ce processus de sélection aléatoire, ce qui permet à Elrond d'avoir un groupe nouvellement et aléatoirement sélectionné qui réussit à engager un nouveau bloc dans le registre à chaque tour.

Le compromis pour cette amélioration repose sur le principe qu'un adversaire ne peut pas s'adapter plus vite que le calendrier du cycle et peut choisir de ne pas proposer le bloc. Une autre amélioration de la sécurité de la source d'aléa serait l'utilisation de fonctions de retard vérifiables (VDF) afin d'empêcher toute possibilité de falsification de la source d'aléa jusqu'à ce qu'il soit trop tard. Actuellement, la recherche sur les VDF est toujours en cours - il n'existe que quelques implémentations VDF fonctionnelles (et mal testées).

3) En plus du facteur de mise en jeu généralement utilisé dans les architectures de PoS comme seul élément de décision, Elrond affine son mécanisme de consensus en ajoutant un facteur de pondération supplémentaire appelé notation. La probabilité que le nœud soit sélectionné dans le groupe de consensus prend en considération à la fois l'enjeu et la notation. La notation d'un proposant de bloc est recalculée à la fin de chaque époque, sauf dans les cas où des coupes devraient se produire, où la diminution réelle de la notation est faite instantanément, ajoutant une autre couche de sécurité en promouvant la méritocratie.

4) Un système modifié de signature multiple du BLS [5] avec 2 cycles de communication est utilisé par le groupe de consensus pour la signature en bloc

5) Elrond envisage une vérification formelle pour les implémentations critiques du protocole (par exemple, le mécanisme de consensus SPoS) afin de valider la justesse de nos algorithmes.

II Aperçu de l'architecture

1 Entités

Il y a deux entités principales dans Elrond : les utilisateurs et les nœuds. Les utilisateurs détiennent tous un nombre (fini) de (Pk/sk) (par exemple dans une ou plusieurs applications de portefeuille). Ils utilisent le réseau Elrond pour déployer des transactions signées, afin de transférer de la valeur, ou pour exécuter des Smarts Contracts. Les utilisateurs sont identifiés par l'une de leurs adresses de compte (dérivées de la clé publique). Les nœuds sont représentés par les terminaux dont est constitué le réseau Elrond, et peuvent être engagés passivement ou activement dans le traitement des tâches. Les validateurs éligibles sont des participants actifs dans le réseau Elrond. Plus précisément, ils sont chargés de l’application du consensus, de l'ajout de blocs, du maintien du statut, en échange de quoi, ils perçoivent des récompenses. Chaque validateur éligible est identifié de manière unique par une clé publique dérivée à la fois de l'adresse où est déposé le collatéral, et aussi de l'identifiant du nœud.

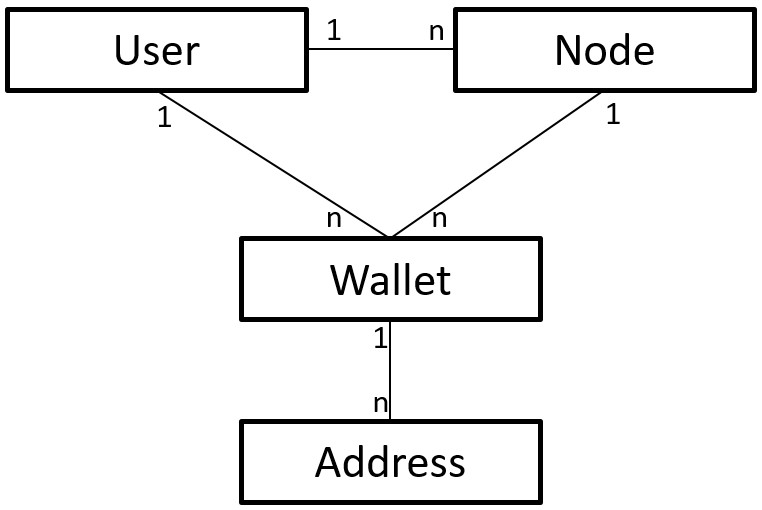


Fig. 1 : Relations entre les entités Elrond

De plus, le réseau est divisé en unités plus petites appelées fragments. Un valideur éligible est attribué à un fragmentsur la base d'un algorithme qui maintient les nœuds uniformément répartis à travers les fragment, en fonction du niveau dans l'arbre. Chaque shard contient un groupe de consensus choisi au hasard. Quiconque propose un bloc est responsable de l'agrégation des transactions dans un nouveau bloc. Les validateurs sont chargés soit de rejeter, soit d'approuver le bloc proposé, de sorte à le valider et à l’inscrire définitivement dans la blockchain.

2 Jeton intrinsèque

La façon dont Elrond autorise l'accès pour l’utilisation de son réseau se fait par le biais de jetons d’utilité intrinsèque appelés Elronds, en abrégé ERD. Tous les coûts liés au traitement des transactions, à l'exécution des « smart contracts » et aux récompenses pour les diverses contributions au réseau seront payés en ERD. Les références aux frais, aux paiements ou aux soldes sont supposées être en ERD.

3. Modèle de menace

Elrond s’appuie sur un modèle contradictoire byzantin, dans lequel au moins 2 n+1 des nœuds éligibles d'un shard sont honnêtes. Le protocole rend acceptable l'existence d'adversaires qui disposent d’un enjeu ou d’une bonne cote, qui puissent retarder ou envoyer des messages conflictuels, qui compromettent d'autres nœuds, qui présentent des bogues ou qui sont de connivence entre eux, mais tant que 2 n+1 des validateurs éligibles dans un shard sont honnêtes/non compromis, le protocole peut aboutir à un consensus.

Le protocole présuppose que les adversaires sont hautement adaptables, sans pour autant pouvoir s'adapter plus rapidement que la durée d'un tour. La puissance de calcul d'un adversaire est limitée, c'est pourquoi les hypothèses cryptographiques consenties par le niveau de sécurité des primitives choisies sont prises fermement à l’intérieur de la classe de complexité des problèmes pouvant être résolus par une machine de Turing en temps polynomial. Le réseau des nœuds honnêtes est supposé former un graphe bien connecté et la propagation de leurs messages se fait dans un temps borné ∆.

Prévention des vecteurs d'attaque

1) Attaques Sybil : atténuées par le verrouillage de la mise en jeu au moment de rejoindre le réseau. Ainsi, la génération des nouvelles identités présente un coût égal à celui de l'enjeu minimal ;

2) Rien en jeu : supprimé de par le besoin en signatures multiples, en plus de celle du proposant, et de par la réduction de l'enjeu. Comparée à la récompense par bloc, l'enjeu verrouillé découragera de tels comportements ;

3) Attaques longue distance : atténuées par notre mécanisme d'élagage, et par l'utilisation d'un groupe de consensus sélectionné au hasard à chaque tour (et pas uniquement un seul proposant) ainsi que par le verrouillage de l'enjeu. De plus, notre algorithme de consensus pBFT garantit l’irrévocabilité ;

4) Les attaques DDoS : le groupe de consensus est échantillonné au hasard à chaque tour (quelques secondes) ; le faible délai rendant le DDoS presque impossible.

Les autres vecteurs d'attaque que nous avons pris en considération sont : attaque par prise de contrôle des fragments, censure des transactions, doubles dépenses, attaques de corruption, etc.

4 Chronologie

Dans le réseau d'Elrond, la chronologie est divisée en époques et en tours. Les époques ont une durée constante, fixée à un jour (peut être modifiée à mesure que l'architecture évoluera), à l'issue de laquelle la réorganisation et l'élagage des fragments sont déclenchés. Les époques sont ensuite divisées en tours, d'une durée fixe. Un nouveau groupe de consensus est sélectionné au hasard par fragment à chaque tour, qui peut valider au maximum un bloc dans le registre du fragment.

Les nouveaux valideurs peuvent rejoindre le réseau en verrouillant leur enjeu, comme présenté au chapitre V.2 - Preuve d'enjeu sécurisée. Ils sont ajoutés au pool de nœuds non attribués de l'époque courante e, sont inscrits sur la liste d'attente d’un fragment au début de l'époque e + 1, mais ne peuvent devenir des valideurs éligibles pour participer au consensus et être récompensés, uniquement à l'époque e + 2 suivante.

Les aspects relatifs à la chronologie sont détaillés dans la section IX.1.

II Travaux connexes

Elrond a été conçu et inspiré par les idées d'Ethereum [6], Omniledger [7], Zilliqa [8], Algorand [3] et ChainSpace [9]. Notre architecture va au-delà de l'état de l'art et peut être considérée comme une augmentation des modèles existants, améliorant les performances tout en se concentrant sur un meilleur équilibre entre sécurité, évolutivité et décentralisation.

1 Ethereum

Une grande partie du succès d'Ethereum [6] peut être attribuée à l'introduction de sa couche d'applications décentralisées par le biais d'EVM [10], Solidity [11] et Web3j [12]. Si les Dapps ont été l'une des caractéristiques essentielles d'Ethereum, l'extensibilité s'est avérée une limitation pressante. Des recherches considérables ont été menées pour résoudre ce problème, mais les résultats ont été négligeables jusqu'à présent. Pour autant, peu d'améliorations prometteuses ont été proposées : Casper [13] prépare une mise à jour qui remplacera le consensus actuel concernant la preuve de travail (proof of work) par une preuve d'enjeu (proof of stake), tandis que les chaînes latérales et les fragments à base de plasma devraient être disponibles dans un avenir proche, ce qui atténuera au moins partiellement le problème d'extensibilité d'Ethereum [14].

Comparé à Ethereum, Elrond élimine les gaspilliages énergétiques et informatiques des algorithmes de PoW en mettant en œuvre un consensus SPoS tout en utilisant le parallélisme du traitement des transactions par fragments.

2 Ominiledger

Omniledger [7] propose un nouveau registre distribué scalable horizontalement qui préserve la sécurité à long terme dans le cadre d'une exploitation libre de permission. Il garantit la sécurité et l'exactitude en utilisant un protocole publique à caractère aléatoire résistant aux biais pour choisir les grands fragments statistiquement représentatifs qui traitent les transactions. Pour engager des transactions de manière atomique entre fragments, Omniledger présente Atomix, un protocole efficace .

Le concept est un protocole de "verrouillage/déverrouillage" à deux phases, piloté par le client, qui garantit que les nœuds peuvent, soit engager complètement une transaction sur des fragments, soit obtenir des "preuves de rejet" pour interrompre et déverrouiller l'état affecté par des transactions partiellement achevées. Omniledger optimise également les performances grâce au traitement parallèle des transactions intra-fragments, à l'élagage du grand livre par le biais de blocs d'état signés collectivement et à la validation à faible latence "faire confiance mais vérifier" pour les transactions de faible valeur. Le consensus utilisé dans Omniledger est une variante de BFT, appelée ByzCoinX, qui augmente les performances et la robustesse contre les attaques de déni de service.

Par rapport à Omniledger, Elrond a une approche adaptative du partage d'État, une sélection aléatoire plus rapide du groupe de consensus et une sécurité améliorée en remplaçant l'ensemble des validateurs après chaque tour (quelques secondes) et non après chaque époque (1 jour).

3 Zilliqa

Zilliqa [8] est la première architecture de partitionnement des transactions permettant au réseau minier de traiter les transactions en parallèle et d'atteindre un débit élevé en divisant le réseau minier en fragments. Plus précisément, sa conception permet un taux de transaction plus élevé à mesure que de nouveaux nœuds rejoignent le réseau. L'essentiel est de veiller à ce que les fragments traitent des transactions différentes, sans chevauchement et donc sans double dépense. Zilliqa utilise le pBFT [15] pour le consensus et le PoW pour établir les identités et prévenir les attaques de type Sybil.

Par rapport à Zilliqa, Elrond repousse les limites du partitionnement en utilisant, non seulement le partitionnement de transaction, mais aussi le partitionnement d'État. Elrond élimine complètement le mécanisme du PoW et utilise le SPoS pour le consensus. Les deux architectures construisent leur propre moteur de contrat intelligent, mais Elrond vise non seulement la conformité EVM, de sorte que le SC écrit pour Ethereum fonctionne de manière transparente sur notre VM, mais vise également à réaliser l'interopérabilité entre les blockchains.

4 Algorand

Algorand [3] propose un grand livre public qui conserve la commodité et l'efficacité des systèmes centralisés, sans les inefficacités et les faiblesses des mises en œuvre décentralisées actuelles. Le chef et l'ensemble des vérificateurs sont choisis au hasard, en fonction de leur signature appliquée à la valeur de la quantité du dernier bloc. Les sélectionnés sont à l'abri des manipulations et imprévus jusqu'au dernier moment. Le consensus repose sur un nouvel accord byzantin de transmission de messages qui permet à la communauté et au protocole d'évoluer sans bifurquer. Par rapport à Algorand, Elrond n'a pas de blockchain unique, mais augmente le débit des transactions grâce aux fragments. Elrond améliore également l'idée de sélection aléatoire d'Algorand en réduisant le temps de sélection du groupe de consensus de plus de 12 secondes à moins d'une seconde, mais suppose que

les adversaires ne peuvent pas s'adapter au cours d'un tour.

Par rapport à Chainspace, où le TPS diminue avec chaque nœud ajouté dans une shard, l'approche d'Elrond n'est pas influencée par le nombre de nœuds dans celle-ci, car le groupe de consensus a une taille fixe. Un point fort de Chainspace est l'approche des contrats intelligents agnostiques au langage, tandis qu'Elrond se concentre sur la construction d'une couche d'abstraction pour la conformité EVM. Les deux projets utilisent des approches différentes pour les fragments d'État afin d'améliorer les performances. Cependant, Elrond va plus loin en anticipant le problème de la taille des blockchains dans les architectures à haut débit et utilise un mécanisme d'élagage efficace. En outre, Elrond présente une plus grande résistance aux changements soudains de la population de nœuds et à la prise de contrôle malveillante des shards en introduisant la redondance des fragment, une nouvelle fonctionnalité pour les chaînes de blocs fragmentées.

IV Scalabilité via partitionnement d’état adaptatif

1 Pourquoi partitionner horizontalement ?

A l’origine, le partitionnement était utilisé dans les bases de données en tant que méthode de distribution des données entre plusieurs machines. Cette technique de passage à l'échelle peut être appliquée à la blockchain pour partitionner les états et le traitement des transactions, de sorte à ce que chaque nœud ne traite qu'une fraction de toutes les transactions en parallèle avec les autres nœuds. Tant qu'il y a un nombre de nœuds suffisant, et qui vérifient chaque transaction afin que le système maintienne une fiabilité et une sécurité élevées, dans ces conditions, le découpage d'une blockchain donnée, en partitions horizontales lui permettra de traiter de nombreuses transactions en parallèle, et donc d'améliorer considérablement le débit des transactions ainsi que l’efficacité des traitements. La promesse du partitionnement est de pouvoir augmenter le débit au fur et à mesure que le réseau de valideurs s'étend, c’est une propriété designée sous le nom de scalabilité horizontale..

2 Types de sharding

Une introduction complète et détaillée [16] met l'accent sur les trois principaux types sharding : sharding de réseau, sharding de transaction, et sharding d’état. Le sharding de réseau gère la façon dont les nœuds sont regroupés en shard et peut être utilisé pour optimiser la communication, car la propagation des messages à l'intérieur d'un shard peut se faire beaucoup plus rapidement que la propagation à l'ensemble du réseau. C'est le premier défi dans chaque approche de sharding et le mécanisme qui fait correspondre les nœuds aux shard doit prendre en considération les attaques possibles d'un attaquant qui prend le contrôle d'un shard spécifique. Le sharding de transaction gère la manière dont les transactions sont mises en correspondance avec les shards où elles seront traitées. Dans un système basé sur des comptes, les transactions pourraient être attribuées à des shards en fonction de l'adresse de l'expéditeur. L'approche la plus difficile est celle du sharding d’état. Contrairement aux mécanismes de découpage en shards décrits précédemment, où tous les nœuds stockent l'état complet, dans les blockchains découpées horizontalement en shards d’état, chaque shard ne maintient qu'une partie de l’état. Chaque transaction traitant des comptes qui se trouvent dans des shards différente, devrait échanger des messages et mettre à jour les états dans des shards différents. Afin d'accroître la résilience aux attaques malveillantes, les nœuds des shards doivent être re-mélangés de temps en temps. Cependant, le déplacement des nœuds entre les shards introduit des coûts de synchronisation, qui correspond au temps nécessaire pour que les nœuds nouvellement ajoutés puissent télécharger le dernier état. Ainsi, il est impératif que seul un sous-ensemble de tous les nœuds soit redistribué à chaque époque, pour éviter les temps d'arrêt pendant le processus de synchronisation.

3 Les tendances du sharding

Certaines propositions sharding tentent de mettre en oeuvre uniquement du sharding de transactions [8] ou uniquement du sharding d'état [17], ce qui augmente le débit de la transaction, soit en forçant chaque nœud à stocker beaucoup de données d'état, soit en étant un superordinateur [2]. Pourtant, plus récemment, au moins une revendication a été faite quant à la réussite de sharding à la fois de transaction et d'état, sans compromettre la puissance de stockage ou de traitement [13].

Mais le sharding pose de nouveaux défis, comme l'attaque par prise de contrôle d'un shard unique, la communication entre les shards, la disponibilité des données et la nécessité d'une couche d'abstraction qui cache les shards. Toutefois, sous réserve que les problèmes susmentionnés soient correctement traités, le sharding d’état apporte des améliorations globales considérables : le débit des transactions augmente de manière significative en raison du traitement parallèle des transactions et les frais de transaction seront considérablement réduits. Deux principaux critères largement considérés comme des obstacles se transformant en autant d’avantages et d’incitations à l'adoption généralisée de la technologie blockchain.

4 Approche d'Elrond sur le sharding

Tout en faisant face à la complexité de la combinaison du sharding de transaction avec le sharding d’état, l'approche d'Elrond a été conçue avec les objectifs suivants en tête :

1) Scalabilité sans affecter la disponibilité : Augmenter ou diminuer le nombre de partitions devrait affecter de manière négligeable un petit nombre des nœuds qui sont à proximité, sans provoquer de temps d'arrêt, ou de les minimiser tout en actualisant les états ;

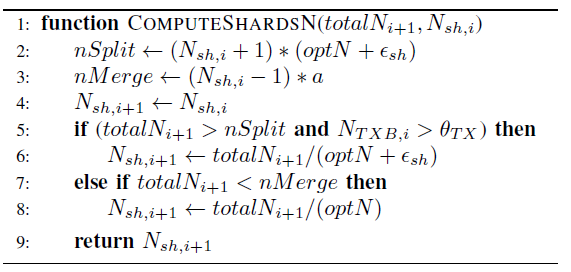
2) Expédition et traçabilité instantanée : La découverte du shard de destination d'une transaction doit être déterministe, triviale à calculer, éliminant le besoin de tours de communication ;

3) Efficacité et adaptabilité : Les partitions doivent être aussi équilibrées que possible à chaque instant.

Description de la méthode

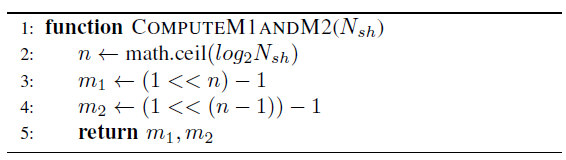
Pour calculer un nombre optimal de shards Nsh dans l'époque ei+1 (Nsh;i+1), nous avons défini un coefficient de seuil pour le nombre de transactions dans un bloc, TX. La variable optN représente le nombre optimal de nœuds dans un shard, sh est un nombre positif et représente la fourchette de variation du nombre de nœuds d'un shard. totalNi est le nombre total de nœuds (validateurs éligibles, nœuds dans les listes d'attente et nouveaux nœuds dans le pool de nœuds) sur toutes les shards de l'époque ei, tandis que NTXB;i est le nombre moyen de transactions dans un bloc sur tous les shards de l'époque ei. Nsh;0 sera considéré comme valant 1.

Le nombre total de shards Nsh;i+1 changera si le nombre de nœuds totalNi du réseau change et si l’utilisation de la blockchain en a besoin : si le nombre de nœuds augmente au-delà d'un seuil nSplit d'une époque à l'autre et que le nombre moyen de transactions par bloc est supérieur au seuil de transactions par bloc NTXB;i > TX ou si le le nombre de nœuds diminue en dessous d'un seuil nMerge comme indiqué dans la fonction ComputeShardsN.

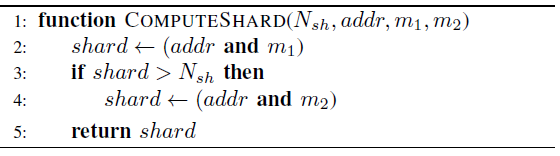


D'une époque à l'autre, il y a une probabilité que le

nombre de shards actifs change. Si cet aspect influence le nombre de shards, n'importe qui peut calculer les deux masques m1 et m2, utilisés dans la distribution des transactions.



L'objectif principal étant d'augmenter le débit au-delà des milliers de transactions par seconde et de diminuer la communication entre les shards, Elrond propose un mécanisme de répartition qui détermine automatiquement les partitions impliqués dans la transaction en cours et achemine la transaction en conséquence. L'expéditeur prendra en considération l'adresse de compte (addr) de l'expéditeur/récepteur de la transaction. Le résultat est le numéro de shard (shard) auquel la transaction sera envoyée.



L'ensemble du système de sharding est basé sur un arbre binaire qui distribue les adresses des comptes, favorise la scalabilité et traite les transitions d'état. Une représentation de l'arbre est visible sur la figure 2.

L'arborescence présentée n'est qu'une représentation logique de l'espace d'adresse du compte utilisé pour une mise en correspondance déterministe ; par exemple, l'attribution des shards, le calcul des noeuds frères, etc. Les feuilles de l'arbre binaire représentent les shards avec leur numéro ID. En partant de la racine (noeud/partition 0), s'il n'y a qu'une seule partition/feuille (a), toutes les adresses de compte sont associées à celui-ci et toutes les transactions seront exécutées là. Plus loin, si la formule de Nsh impose la nécessité de 2 shards (b), l'espace d'adresse sera divisé en parties égales, en fonction des derniers bits de l'adresse.

Parfois, l'arbre peut aussi se déséquilibrer (c) si Nsh

n'est pas une puissance de 2. Ce cas n'affecte que les feuilles du 6ème niveau. La structure redeviendra équilibrée lorsque le nombre de shards atteindra de nouveau une puissance de 2.

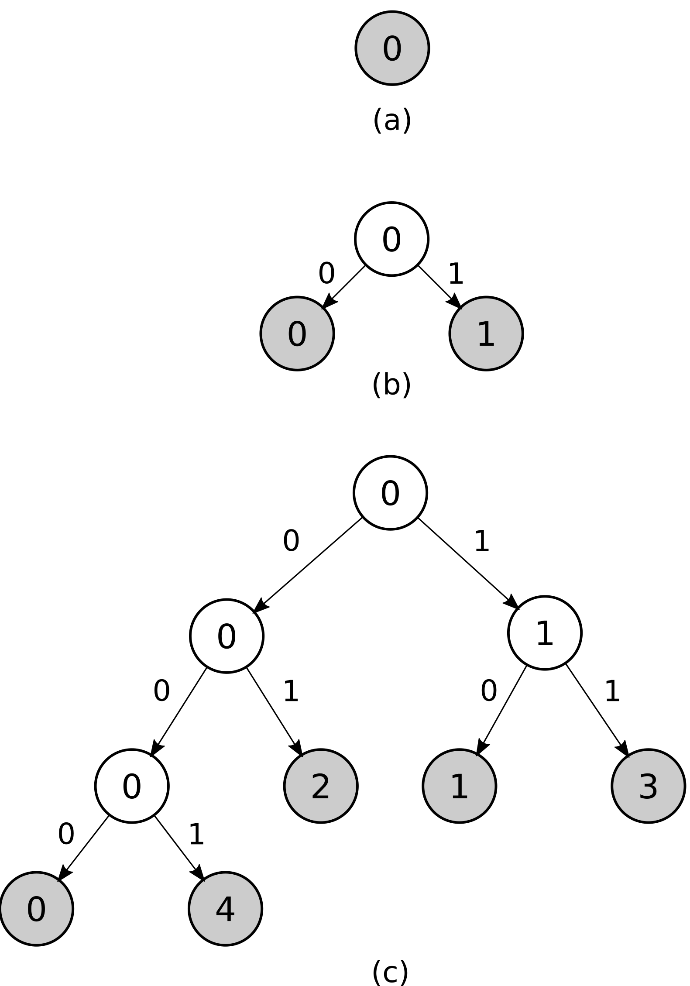


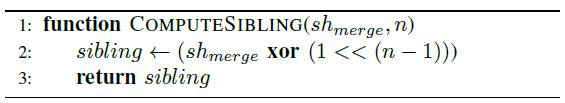
Fig. 2 : Exemple d'une structure arborescente de partition

Lorsque l'arbre binaire n’est plus équilibré, ceci implique que les shards situés au niveau le plus bas occupent la moitié de l'espace d'adressage des nœuds d'un shard située au niveau du dessus, on peut donc affirmer que les nœuds actifs attribués à ces shards percevront des revenus plus faibles - les récompenses de bloc ne sont pas affectées. Toutefois, ce problème est résolu en distribuant un tiers de chaque nœud de shard de façon aléatoire à travers chaque époque (détaillée dans la Chronologie ) et présentant une répartition équilibrée des nœuds selon le niveau de l'arbre.

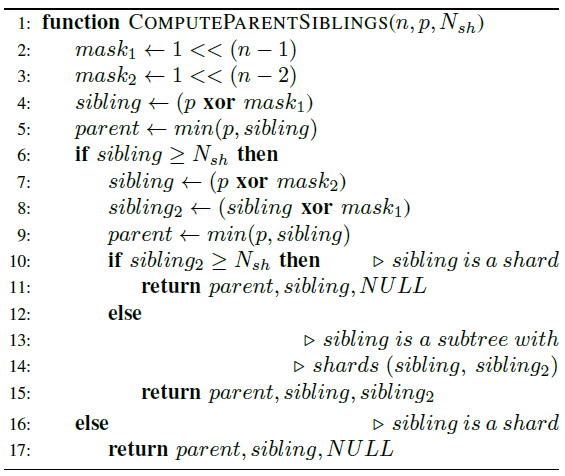
En regardant l'arbre, à partir de n'importe quelle feuille et en progressant le long des branches en direction de la racine, l'encodage des branches correspond aux n derniers bits des adresses de compte qui feront traiter, par cette feuille/shard, leurs transactions d'origine associées. En allant dans l'autre sens, de la racine vers la feuille, l'information est liée au développement de la structure, shards frères, du shard parent d'où ils se séparent. À l'aide de cette hiérarchie, le shard qui se divise lorsque Nsh augmente ou les shards qui fusionnent lorsque Nsh diminue peuvent être facilement calculés. L'ensemble du mécanisme de sharding d'état bénéficie de cette structure en gardant toujours l'adresse et l'état associé dans le même shard..

Connaissant Nsh, n'importe quel nœud peut suivre le processus de redistribution sans avoir besoin de communication. L'allocation des ID pour les nouveaux shards est incrémentale et la réduction du nombre de shards implique que les shards les plus élevés seront supprimés. Par exemple, lorsque vous allez de Nsh à Nsh-1, deux shards seront fusionnés, le shards à supprimer est

le shard numéroté le plus élevé (Shmerge=NSH-1). Trouver le numéro de shard avec lequel shmerge sera fusionné est trivial. Selon la structure de l'arbre, le shard résultant aura le numéro du frère :



Pour la redondance des shards, la traçabilité des transitions d'état et le passage à l'échelle rapide, il est important de déterminer le frère et le parent d'un shard générique avec le nombre p :



redondance des shard

33 Sur la blockchain, le partage d'état est exposé aux échecs de shards lorsqu'il y a un nombre insuffisant de nœuds en ligne dans un shard ou que la distribution est localisée géographiquement. Dans le cas peu probable où un shard échoue (soit le shard ne peut pas être contacté - tous les nœuds sont hors-lignes ou un consensus ne peut pas être atteint - plus d'un des nœuds ne répond pas), il y a un risque élevé que l'architecture entière ne repose que sur des nœuds super-pleins [2], qui téléchargent entièrement chaque bloc de chaque shard, vérifiant entièrement tout. Comme le montre la Fig. 3, notre protocole dispose d'un mécanisme de protection qui introduit un compromis dans la structure de maintien de l'état en imposant aux shards du dernier niveau de l'arbre de maintenir également l'état de leurs frères. Ce mécanisme réduit la communication et élimine le bootstrapping lorsque des shards frères fusionnent puisqu'ils ont déjà les données.

Changement de contexte

Pour préserver la sécurité dans blockchains publiques appliquant le sharding, le changement de contexte devient crucial [7]. Cela fait référence à la réaffectation des nœuds actifs entre les shards sur un intervalle de temps fixe selon certains critères aléatoires. Dans l'approche d'Elrond, le changement de contexte apporte une amélioration de la sécurité, mais augmente également la complexité nécessaire pour maintenir la cohérence entre plusieurs états. La transition d'état laisse la plus grande empreinte sur les performances puisque le mouvement des nœuds actifs nécessite de resynchroniser l'état, la blockchain et les transactions en même temps que les nœuds éligibles dans le nouveau shard.

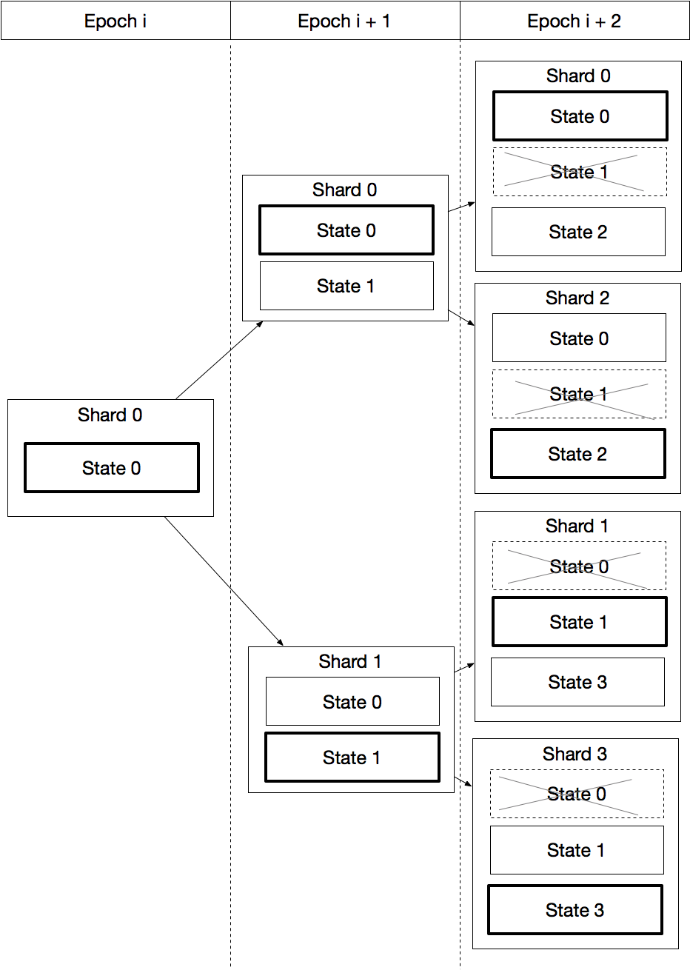


Fig. 3: Redondance des shards à travers les époques

Au début de chaque époque, afin de maintenir la viabilité, seul moins d'un de ces nœuds sera uniformément redistribué entre les shards. Ce mécanisme est très efficace contre la formation de groupes malveillants.

5 Chaîne de notarisation (Meta)

Toutes les opérations relatives au réseau et aux données globales (nœud rejoignant le réseau, nœud quittant le réseau, calcul des listes de validateurs éligibles, affectation des nœuds aux listes d'attente des shards, accord par consensus sur un bloc dans un shard spécifique, les contestations des blocs non valides) seront notariées dans la métachaîne. Le consensus de la métachaîne est géré par un shard différent qui communique avec tous les autres shard et facilite les opérations entre shards. À chaque tour de chaque époque, la métachaîne reçoit les en-têtes de bloc des autres shards et, si nécessaire, les preuves des contestations des blocs invalides. Ces informations seront regroupées en blocs sur la métachaîne sur lesquels un consensus doit être exécuté. Une fois les blocs validés dans le groupe de consensus, les shards peuvent demander des informations sur les blocs, les miniblocs (voir chapitre VII), les validateurs éligibles, les nœuds des listes d'attente, etc. afin de traiter en toute sécurité les transactions entre shards. De plus amples informations sur l'exécution des opérations croisées entre shards, la communication entre les shards et la métachaîne seront présentées au chapitre VII Traitement des transactions croisées entre shards.

V Consensus par preuve d'enjeu sécurisée

1 Analyse du consensus

Le premier algorithme de consensus de blockchain basé sur la Preuve de Travail (Proof of Work = PoW), est utilisé dans Bitcoin, Ethereum et d'autres plates-formes blockchain. Dans la Preuve de Travail, chaque nœud est nécessaire pour résoudre un puzzle mathématique (difficile à calculer mais facile à vérifier). Et le premier nœud qui termine le puzzle recueille la récompense [18]. Les mécanismes de Preuve de Travail permettent d'éviter les doubles dépenses, les attaques DDoS et Sybil au prix d'une consommation d'énergie élevée.

La Preuve d'Enjeu (PoS = Proof of Stake) est un mécanisme de consensus nouveau et plus efficace proposé comme une alternative à la consommation intensive d'énergie et de puissance de calcul dans les mécanismes de consensus de Preuve de Travail (PoW). La PoS peut être trouvée dans de nombreuses nouvelles architectures comme Cardano [19] et Algorand [3] et peut être dans la prochaine version d'Ethereum. Dans la PoS, le nœud qui propose le bloc suivant est sélectionné par une combinaison d'enjeux (richesse), de tirage aléatoire et/ou d'âge. Cela atténue le problème de l'énergie du PoW mais pose également deux problématique importantes sur la table : l'attaque "Rien à perdre" et un risque de centralisation plus élevé.

Proof of Meme, tel qu'il est envisagé dans Constellation [20], est un algorithme basé sur la participation historique du nœud au réseau. Son comportement est stocké dans une matrice de pondération dans la blockchain et prend en charge les changements au fil du temps. Il permet également aux nouveaux nœuds de gagner la confiance des utilisateurs en se forgeant une réputation. Le principal inconvénient des attaques Sybil est atténué par l'algorithme NetFlow.

La preuve d'autorité déléguée (Delegated Proof of Stake - DPoS), que l'on trouve dans Bitshares [21], Steemit [22] et EOS [23], est un hybride entre la preuve d'autorité et la preuve d'enjeu, dans lequel les quelques nœuds responsables du déploiement de nouveaux blocs sont élus par les parties prenantes. Bien que présentant un débit élevé, le modèle est sensible aux problèmes sociaux spécifiques à l'homme tels que la corruption. De plus, un petit nombre de délégués rend le système vulnérable aux attaques DDoS et à la centralisation.

2 Preuve d'enjeu sécurisée (SPoS = Secure Proof of Stake)

L'approche d'Elrond en matière de consensus consiste à combiner la sélection aléatoire des validateurs, l'éligibilité à travers l'enjeu et de la notation, avec une dimension optimale pour le groupe de consensus. L'algorithme est décrit dans les étapes ci-dessous :

1) Chaque nœud ni est défini comme un tuple de clé publique (P k), de notation (par défaut 0) et l'enjeu verrouillé. Si ni souhaite participer au consensus, il doit d'abord s'enregistrer par le biais d'un contrat intelligent, en envoyant une transaction qui contient un montant égal à l'enjeu minimal requis et d'autres informations (P ks, une clé publique dérivée de P k et un nodeid qui sera utilisé pour le processus de signature afin de ne pas utiliser une adresse réelle de portefeuille).

2) Le nœud ni rejoint le pool de nœuds et attend l'affectation des shards à la fin de l'époque e en cours. Le mécanisme d'affectation des shards crée un nouvel ensemble de nœuds contenant tous les nœuds qui se sont joints à l'époque e et tous les 33 nœuds qui doivent être remélangés (moins de 1 de chaque shards). Tous les nœuds de cet ensemble seront réaffectés aux listes d'attente de shards. Wj représente la liste d'attente de shards de j et Nsh représente le nombre de shards. Un nœud possède également une clé secrète sk qui, par nature, ne doit pas être rendue publique.

ni = (P ki, ratingi, stakei)

ni Wj, 0 ≤ j < Nsh

3) À la fin de l'époque il s'est joint, le nœud sera déplacé vers la liste des nœuds éligibles (Ej) d'un shard j, où e est l'époque courante.

ni Wj, e−1 → ni 6Wj, e, ni Ej, e

4) Chaque nœud de la liste Ej peut être sélectionné dans le cadre d'un groupe de consensus dimensionné de manière optimale (en termes de sécurité et de communication), par une fonction déterministe, basée sur le caractère aléatoire de la source ajoutée au bloc précédent, le tour r et un ensemble de paramètres de variation. Le nombre aléatoire, connu de tous les nœuds du shard par le biais d'un protocole de bavardage, ne peut être prédit avant que le bloc ne soit effectivement signé par le groupe de consensus précédent. Cette propriété en fait une bonne source aléatoire et empêche les attaques malveillantes hautement adaptatives. Nous définissons une fonction de sélection pour renvoyer l'ensemble des nœuds choisis (groupe de consensus) Nchosen, le premier étant le proposant du bloc, qui prend les paramètres suivants : E, r et sigr-1 - la signature du bloc précédent.

Nchoisi = f (E, r, sigr−1), où Nchoisi E

5) Le bloc sera créé par le proposant de bloc et les validateurs le cosigneront sur la base du protocole "practical Byzantine Fault Tolerance" (PbFt).

6) Si, pour une raison quelconque, le proposant de bloc n'a pas créé de bloc pendant le créneau horaire qui lui était attribué (malveillant, hors ligne, etc.), le tour r sera utilisé en conjugaison avec la source aléatoire du dernier bloc pour sélectionner un nouveau groupe de consensus.

Si le proposant du bloc courant agit de manière malveillante, les autres membres du groupe émettront un retour négatif pour dégrader son classement, diminuant ou même annulant les chances que ce nœud particulier soit à nouveau sélectionné. La fonction de rétroaction pour le proposant de bloc (ni) dans le tour numéro r, avec le paramètre ratingModifier ∈ Z est calculée comme suit :

feedbackfunction = ff (ni, notationModificateur, r)

< problème >

ni = (P ki, ratingi, stakei)

ni Wj, 0 ≤ j < Nsh

3) À la fin de l'époque il s'est joint, le nœud sera déplacé vers la liste des nœuds éligibles (Ej) d'un shard j, où e est l'époque courante.

ni Wj, e−1 → ni 6Wj, e, ni Ej, e

4) Chaque nœud de la liste Ej peut être sélectionné dans le cadre d'un groupe de consensus dimensionné de manière optimale (en termes de sécurité et de communication), par une fonction déterministe, basée sur le caractère aléatoire de la source ajoutée au bloc précédent, le tour r et un ensemble de paramètres de variation. Le nombre aléatoire, connu de tous les nœuds du shard par le biais d'un protocole de bavardage, ne peut être prédit avant que le bloc ne soit effectivement signé par le groupe de consensus précédent. Cette propriété en fait une bonne source aléatoire et empêche les attaques malveillantes hautement adaptatives. Nous définissons une fonction de sélection pour renvoyer l'ensemble des nœuds choisis (groupe de consensus) Nchosen, le premier étant le proposant du bloc, qui prend les paramètres suivants : E, r et sigr-1 - la signature du bloc précédent.

Nchoisi = f (E, r, sigr−1), où Nchoisi E

5) Le bloc sera créé par le proposant de bloc et les validateurs le cosigneront sur la base du protocole "practical Byzantine Fault Tolerance" (PbFt).

6) Si, pour une raison quelconque, le proposant de bloc n'a pas créé de bloc pendant le créneau horaire qui lui était attribué (malveillant, hors ligne, etc.), le tour r sera utilisé en conjugaison avec la source aléatoire du dernier bloc pour sélectionner un nouveau groupe de consensus.

Si le proposant du bloc courant agit de manière malveillante, les autres membres du groupe émettront un retour négatif pour dégrader son classement, diminuant ou même annulant les chances que ce nœud particulier soit à nouveau sélectionné. La fonction de rétroaction pour le proposant de bloc (ni) dans le tour numéro r, avec le paramètre ratingModifier ∈ Z est calculée comme suit :

feedbackfunction = ff (ni, notationModificateur, r)

Lorsque RatingModifier < 0, une coupure se produit de sorte que le nœud ni perde sa mise.

Le protocole de consensus reste sûr face aux attaques DDoS dans la mesure où il peut s'appuyer sur un nombre élevé de validateurs possibles existants dans la liste E (centaines de nœuds) et dans la mesure ou il n'y a aucun moyen de prédire l'ordre des validateurs avant qu'ils ne soient sélectionnés.

Pour réduire le cout additionnel de communication qui accompagne un nombre accru de shards, un consensus sera exécuté sur un bloc composite. Ce bloc composite est formé par :

• Le bloc registre : le bloc à ajouter dans le registre du shard, comportant toutes les transactions intra shard ainsi que toutes les transactions inter shards pour lesquelles une preuve de confirmation a été reçue;

• Mini-blocs multiples : chacun d'eux détenant des transactions inter shards pour un shard différent ;

Le consensus ne sera exécuté qu'une seule fois, sur le bloc composite contenant les transactions à la fois intra- et inter-shards. Une fois le consensus obtenu, l'en-tête de bloc de chaque shard est envoyé à la métachaîne pour notarisation

VI La couche cryptographique

1 Analyse des signatures

Les signatures numériques sont des cryptographiques primitif utilisées pour assurer la sécurité de l'information en offrant plusieurs propriétés comme l'authentification du message, l'intégrité des données et la non-répudiation [24].

La plupart des schémas utilisés pour les plates-formes blockchaine existantes reposent sur le problème du logarithme discret (DL) : fonction d’exponentialliter unidirectionnelle y → αymod p. Il est scientifiquement prouvé que le calcul du logarithme discret avec base est difficile [25]. La cryptographie à courbe elliptique (ECC) utilise un groupe cyclique de points au lieu d'un groupe cyclique d'entiers. Ce système réduit l'effort de calcul, de sorte que pour des longueurs de clé de seulement 160 à 256 bits, l'ECC offre le même niveau de sécurité que RSA, Elgamal, DSA et d'autres fournissent pour des longueurs de clé de 1024 à 3072 bits (voir tableau 1 [24]).

La raison pour laquelle l'ECC offre un niveau de sécurité similaire pour des longueurs de paramètres beaucoup plus petites est que les attaques existantes sur des groupes de courbes elliptiques sont plus faibles que les attaques DL entières existantes, la complexité de ces algorithmes nécessitant une moyenne √p des étapes de résolution. Cela signifie qu'une courbe elliptique utilisant un p premier de 256 bits offre en moyenne un niveau de sécurité de

2128 étapes nécessaires pour le briser [24].

Ethereum et Bitcoin utilisent tous deux la cryptographie par courbe, avec l'algorithme de signature ECDSA. La sécurité de l'algorithme est très dépendante du générateur de nombres aléatoires, car si le générateur ne produit pas un nombre différent à chaque requête, la clé privée peut être perdue [26].

Un autre système de signature numérique est l'EdDSA, une variante de Schnorr basée sur des courbes Edwards qui permettent un calcul rapide [27]. Contrairement à l'ECDSA, il est prouvé qu'il est non malléable, ce qui signifie qu'à partir d'une simple signature, il est impossible de trouver un autre ensemble de paramètres qui définisse le même point sur la courbe elliptique [28], [29]. En outre, l'EdDSA n'a pas besoin d'un générateur de nombres aléatoires car il utilise un nonce, calculé comme le hachage de la clé privée et du message, ce qui permet d'éviter le vecteur d'attaque d'un générateur de nombres aléatoires brisé qui peut révéler la clé privée.

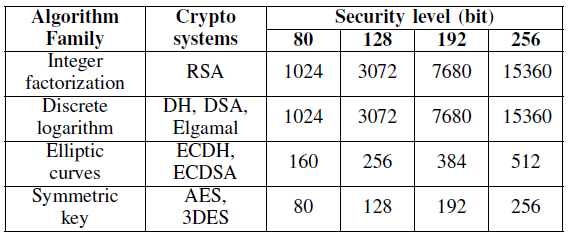


TABLEAU 1 : Longueur des bits des algorithmes à clé publique pour différents niveaux de sécurité

Les variantes de signature de Schnorr attirent de plus en plus l'attention [8], [30] en raison de leur capacité multi-signature native et de leur sécurité prouvée dans le modèle de l'oracle aléatoire [31]. Un système multi-signature est une combinaison d'un algorithme de signature et de vérification, où plusieurs signataires, chacun avec ses propres clés privées et publiques, peuvent signer le même message, produisant une seule signature [32], [33]. Cette signature peut ensuite être vérifiée par un vérificateur qui a accès au message et aux clés publiques des signataires. Une méthode sous-optimale consisterait à demander à chaque nœud de calculer sa propre signature et de concaténer ensuite tous les résultats en une seule chaîne. Cependant, une telle approche est irréalisable car la taille de la chaîne générée augmente avec le nombre de signataires. Une solution pratique consisterait à agréger les résultats en une seule signature de taille fixe, indépendamment du nombre de participants. De tels systèmes ont été proposés à plusieurs reprises, la plupart d'entre eux étant susceptibles de faire l'objet d'attaques de type "rogue-key" (annulation). Une solution à ce problème serait d'introduire une étape où chaque signataire doit prouver la possession de la clé privée associée à sa clé publique [34].

Bellare et Neven [35] (BN) ont proposé un système sécurisé à signatures multiples sans preuve de possession, dans le modèle à clé publique simple, sous l'hypothèse du logarithme discret [31]. Les participants s'engagent d'abord sur leur part de Ri en proposant son hachage à tous les autres signataires afin qu'ils ne puissent pas en calculer une fonction. Chaque signataire calcule un défi différent pour sa signature partielle. Toutefois, ce système sacrifie l'agrégation de la clé publique. Dans ce cas, la vérification de la signature agrégée nécessite la clé publique de chaque signataire. Un article récent de Gregory Maxwell et al (29) propose un autre schéma multi-signature dans le modèle de clé publique simple (36), sous l'hypothèse d'un "logarithme plus discret" (OMDL). Cette approche améliore le schéma précédent [35] en réduisant les cycles de communication de 3 à 2, réintroduisant l'agrégation de clés avec un coût de complexité plus élevé.

BLS [4] est un autre schéma de signature intéressant, issu de l'appariement de Weil, qui fonde sa sécurité sur l'hypothèse de la différence de calcul de Hellman sur certaines courbes elliptiques et génère de courtes signatures. Il possède plusieurs propriétés utiles comme la vérification par lots, l'agrégation de signatures, l'agrégation de clés publiques, ce qui fait du BLS un bon candidat pour les systèmes à seuil et à signatures multiples.

Dan Boneh, Manu Drijvers et Gregory Neven ont récemment proposé un système multi-signature du BLS [5], utilisant des idées issues des travaux précédents [35], [30] pour fournir au système des défenses contre les attaques de clés malhonnêtes. Le système permet une vérification efficace avec seulement deux paires nécessaires pour vérifier une signature multiple et sans aucune preuve de la connaissance de la clé secrète (fonctionne dans le modèle de clé publique). Un autre avantage est que la multi-signature peut être créée en seulement deux cycles de communication.

Pour des raisons de traçabilité et de sécurité, un consensus basé sur un ensemble réduit de validateurs exige la clé publique de chaque signataire. Dans ce contexte, notre analyse conclut que le schéma multi-signature le plus approprié pour la signature en bloc à Elrond est le BLS multi-signature [5], qui est globalement plus rapide que les autres options en raison de seulement deux cycles de communication.

2 La signature de bloc dans Elrond

Pour la signature de blocs, Elrond utilise une cryptographie de courbe basée sur le schéma multi-signature BLS sur le groupe bilinéaire bn256, qui met en œuvre l'appariement Optimal Ate sur une courbe Barreto Naehrig 256 bits. L’appariement bilinéaire est défini comme suit:

e : g0 × g1 → gt (1)

où g0, g1 et gt sont des courbes elliptiques d'ordre premier p définies par bn256, et e est une carte bilinéaire (c'est-à-dire une fonction d'appariement). Soit G0 et G1 sont les générateurs de g0 et g1. Soit également H0 une fonction de hachage qui produit des points sur la courbe g0 :

H0 : M → g0 (2)

où M est l'ensemble de tous les messages binaires possibles, quelle que soit leur longueur. Le système de signature utilisé par Elrond utilise également une seconde fonction de hasing, dont les paramètres sont connus de tous les signataires :

H1 : M → Zp (3) Chaque signataire i a sa propre paire de clés privées/publiques (SKI, PKI),

où ski est choisi au hasard dans Zp. Pour chaque paire de clés, la propriété P ki = ski - G1 est retenue.

Soit L = P k1, P k2, ..., P kn est l'ensemble des clés publiques de tous les signataires possibles au cours d'un tour spécifique qui, dans le cas d'Elrond, est l'ensemble des clés publiques de tous les nœuds du groupe de consensus. Les deux étapes du processus de signature en bloc sont présentées ci-dessous : la signature et la vérification.

Signature pratique - Cycle 1

Le chef du groupe de consensus crée un bloc avec les transactions, puis signe et diffuse ce bloc aux membres du groupe de consensus.

Signature pratique - Cycle 2

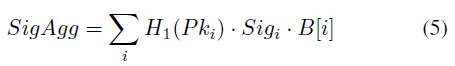
Chaque membre du groupe de consensus (y compris le chef) qui reçoit le bloc doit le valider, et s'il le trouve valide, il le signe avec le BLS et envoie ensuite la signature au chef :

Sigi = ski ∗ H0(m) (4)

Où Sigi est un point sur g0

Signature pratique - Cycle 3

Le chef du groupe attend de recevoir les signatures pendant une période déterminée. S'il n'en reçoit pas au moins 2 - n + 1 signatures dans ce délai, le cycle de consensus est avorté. Mais si le leader reçoit 2 - n + 1 ou plus de signatures valables, il les utilise pour générer la signature agrégée :



Où SigAgg est un point sur g0

Le leader ajoute ensuite la signature agrégée au bloc avec les signataires sélectionnés en bitmap B, où un 1 indique que le signataire correspondant dans la liste L a vu sa signature ajoutée à la signature agrégée SigAgg.

Vérification pratique

Compte tenu de la liste des clés publiques L, de la bitmap des signataires B, de la signature agrégée SigAgg et d'un message m (bloc), le vérificateur calcule la clé publique agrégée :

P kAgg = X H1(P ki) · P ki · Bi (6)

i

Le résultat, P kAgg, est un point sur g1. La vérification finale est l'endroit où e est la fonction d'appariement.

e(G1, SigAgg) == e(P kAgg, H0(m)) (7)

VII Exécution inter shards

Pour un exemple approfondi de la manière dont les opérations inter shards sont exécutées et illustrer comment la communication entre les shards et la métachaîne se déroulent, nous simplifierons l'ensemble du processus pour ne mettre en jeu que deux shards adossés à la métachaîne. En supposant qu'un utilisateur génère une transaction à partir de son portefeuille, qui a une adresse dans le shard 0 et souhaite envoyer des ERD à un autre utilisateur qui a un portefeuille avec une adresse dans le shard 1, les étapes décrites dans la figure 4 sont nécessaires pour traiter la transaction inter shards. Comme mentionné dans le chapitre V - Consensus via preuve d'enjeu sécurisée, la structure des blocs est représentée par un en-tête de bloc qui rassemble des informations sur le bloc (nombre ad hoc du bloc = "nonce" du bloc, tour, proposant, horodatage des validateurs, etc.), et une liste de miniblocs de chaque shard à l'intérieur duquel sont contenues les transactions en question. Chaque minibloc contient toutes les transactions qui ont soit l'émetteur dans le shard actuel et le récepteur dans un autre shard, soit l'émetteur dans un shard différent et la destination dans le shard actuel. Dans notre cas, pour un bloc dans le shard 0, il y aura normalement 3 miniblocs :

• miniblock 0 : contenant les transactions intra shard pour le shard 0

• miniblock 1 : contenant les transactions inter shards avec l'expéditeur dans le shard 0 et la destination dans le shard 1

• miniblock 2 : contenant les transactions inter shards avec l'expéditeur dans le shard 1 et la destination dans le shard 0. Ces transactions ont déjà été traitées dans le shard de l'expéditeur 1 et seront également finalisées après le traitement dans le shard courant.

Il n'y a pas de limite sur le nombre de miniblocs avec le même expéditeur et le même récepteur dans un bloc. Ce qui signifie que plusieurs miniblocs avec le même expéditeur et le même récepteur peuvent apparaître dans le même bloc.

1 Traitement

Actuellement, l'unité atomique de traitement inter shard est le minibloc : soit toutes les transactions du minibloc sont traitées en même temps, soit aucune, et l'exécution du minibloc sera retentée au tour suivant.

Notre stratégie de transactions inter shards utilise un modèle asynchrone. La validation et le traitement sont effectués d'abord dans le shard de l'expéditeur, puis dans le shard du destinataire. Les transactions sont d'abord expédiées dans le shard de l'expéditeur, car il peut valider entièrement toute transaction initiée à partir du compte dans ce shard - principalement le solde courant. Ensuite, dans le tesson du récepteur, les nœuds n'ont besoin que de la preuve d'exécution offerte par la métachaîne, font la vérification de la signature et vérifient l'absence d'attaque par rediffusion et enfin mettent à jour le solde pour le récepteur, en ajoutant le montant de la transaction.

Le shard 0 traite à la fois les transactions intra shard dans le minibloc 0 et un ensemble de transactions inter shard qui ont des adresses du shard 1 comme récepteur dans le minibloc 1. L'en-tête du bloc et les miniblocs sont envoyés à la métachaîne. La métachaîne notifie le bloc du shard 0, en créant un nouveau bloc de métachaîne (metablock) qui contient les informations suivantes sur chaque minibloc : ID du shard émetteur, ID du shard récepteur, hachage du minibloc.

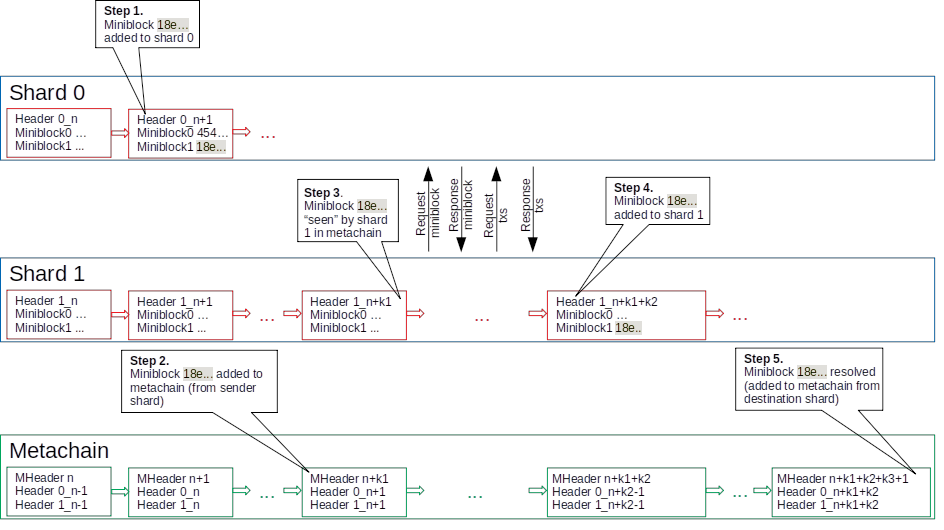


Figure 4 : Traitement des transactions inter shards

Le shard 1 récupère le hachage du minibloc 1 dans le métabloc, demande le minibloc du shard 0, analyse la liste des transactions, requête les transactions manquantes (le cas échéant), exécute le même minibloc 1 dans le shard 1 et envoie au bloc résultant de la métachaîne. Après la notarisation, l'ensemble des transactions inter shards peut être considéré comme finalisée.

Le diagramme suivant montre le nombre de tours nécessaires pour qu'une transaction soit finalisée. Les tours sont comptabilisés entre la première inclusion dans un minibloc et la notarisation du dernier minibloc.

VIII Smart contracts

L'exécution de smart contracts est un élément clé pour toutes les futures architectures de chaînes de blocs. La plupart des solutions existantes évitent de clarifier correctement les dépendances entre les transactions et les données. Ce contexte conduit aux deux scénarios suivants :

1) Lorsqu'il n'y a pas de corrélation directe entre les transactions des smart contracts, comme le montre la figure 5, toute architecture peut utiliser un ordonnancement qui ne garantit pas l'ordre. Cela signifie qu'il n'y a pas de contraintes supplémentaires ni sur le moment, ni sur le lieu (shard) où un smart contract est exécuté.

2) Le second scénario fait référence au parallélisme induit par les transactions qui impliquent des smart contracts corrélés [37]. Ce cas, illustré à la figure 6, ajoute une pression supplémentaire sur les performances et augmente considérablement la complexité. Fondamentalement, il doit y avoir un mécanisme permettant de s'assurer que les contrats sont exécutés dans le bon ordre et au bon endroit (shard). Pour couvrir cet aspect, le protocole Elrond propose une solution qui assigne et déplace le smart contract vers le shard même qui maintient ses dépendances statiques. De cette façon, la plupart, voire la totalité des appels Smart Contracts auront des dépendances réduites au même shard et aucun verrouillage/déverrouillage inter shards ne sera nécessaire.

Elrond se concentre sur la mise en œuvre de la machine virtuelle Elrond, un moteur conforme à la norme EVM. La conformité EVM

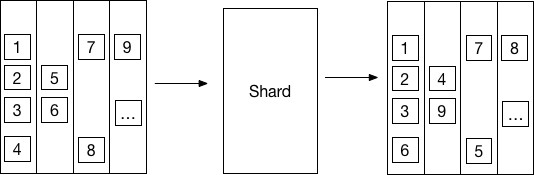


Fig. 5: Traitement indépendant des transactions dans le cadre de smart contracts simples qui peuvent être exécutés sans garantir l'ordre

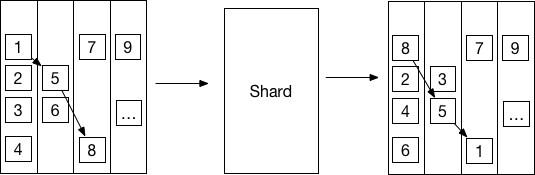


Figure 6 : Mécanisme pour les smart contracts corrélés qui peuvent uniquement être exécutés séquentiellement

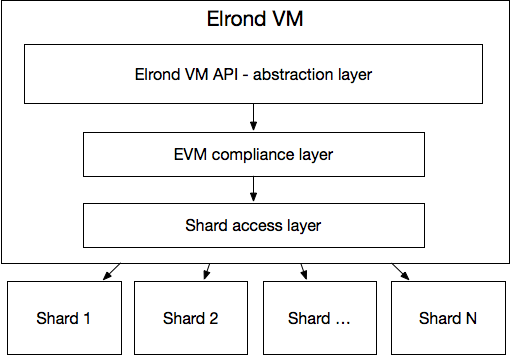


Fig. 7 : Couche d'abstraction pour les smart contracts

est extrêmement importante à des fins d'adoption, en raison du grand nombre de smart contracts construits sur la plateforme d'Ethereum.

L'implémentation de la machine virtuelle Elrond masquera l'architecture sous-jacente isolant les développeurs de smart contracts des systèmes internes assurant ainsi une couche d'abstraction appropriée, comme illustré à la figure 7.

Dans Elrond, l'interopérabilité entre chaînes peut être mise en œuvre en utilisant un mécanisme d'adaptation au niveau de la machine virtuelle, comme le propose Cosmos [38]. Cette approche nécessite des adaptateurs spécialisés et un moyen de communication externe entre les adaptateurs Smart Contracts pour chaque chaîne qui interopérera avec Elrond. L'échange de valeur sera opéré à l'aide de certains smart contracts spécialisés agissant en tant que dépositaires, aptes à assurer la garde des jetons natifs de la chaîne ainsi adaptée et d'émettre des jetons natifs Elrond.

1 infrastructure de machines virtuelles

Elrond construit son infrastructure de Machines Virtuelles au dessus du Framework K, qui est un framework sémantique exécutable permettant de définir des langages de programmation, de calculs, ainsi que des systèmes de types ou des outils d'analyse formels [39].

Le plus grand avantage de l'utilisation du Framework K c'est qu'avec lui, les langages des smarts contract peuvent être définis de manière non- ambiguë, ce qui élimine le risque de comportements non spécifiés et de bogues difficiles à détecter.

Le Framework K est exécutable, en ce sens que les spécifications sémantiques des languages peuvent être utilisé directement comme des interpréteurs de langages opérationnels. Plus précisément, on peut soit exécuter des programmes en fonction des spécifications en utilisant directement l'implémentation coeur du Framework K, soit générer un interpréteur dans plusieurs langages de programmation différents. Ces derniers sont également appelés "backends". Pour des raisons de rapidité d'exécution et de facilité d'interopérabilité, Elrond utilise son propre backend du Framework K, qui est construit sur mesure.

2 Languages des Smart Contracts

Un grand avantage du framework K est que l'on peut générer un interpréteur pour n'importe quelle langue définie en K, sans avoir besoin de programmation supplémentaire. Cela signifie également que les interpréteurs produits de cette manière sont "corrects par construction".

Il existe déjà plusieurs langages de smart contracts spécifiés dans le Framework K, ou dont les spécifications sont en cours d'élaboration. Le réseau Elrond prendra en charge trois langages de bas niveau : IELE VM, KEVM et WASM.

- IELE VM est un langage de niveau intermédiaire, dans le style de LLVM, mais adapté à la chaîne de blocs. Il a été construit directement en K, aucune autre spécification ou implémentation de celui-ci n'existe en dehors du framework K [40]. Son but est d'être lisible par l'homme, rapide, et de surmonter certaines limitations de l'EVM. Elrond utilise une version légèrement modifiée d'IELE - la plupart des changements sont liés à la gestion des adresses des comptes. Les développeurs de smart contracts peuvent programmer directement en IELE, mais la plupart choisiront de coder en Solidity et d'utiliser ensuite un compilateur Solidity to IELE, comme on peut le voir sur la figure 8.

- KEVM est une version de la machine virtuelle Ethereum (EVM), écrite en K [41]. Certaines vulnérabilités de l'EVM sont corrigées dans la version en K, et les fonctionnalités vulnérables sont entièrement écartées.

- Le Web Assembly (WASM) est un format d'instruction binaire, destiné aux machines virtuelles à mémoire de pile, qui peut être utilisé pour exécuter des smart contracts. Une infrastructure WASM permet aux développeurs d'écrire des smart contracts en C/C++, Rust, C#, et autres

Avoir une spécification linguistique et générer l'inter-préteur n'est que la moitié du défi. L'autre moitié consiste à intégrer l'interprète généré au réseau Elrond. Nous avons construit une interface VM commune, qui nous permet de connecter n'importe quelle VM à un nœud Elrond, comme le montre la figure 9. Chaque VM dispose alors d'un adaptateur qui met en œuvre cette interface. Chaque contrat est enregistré comme bytecode de la VM pour laquelle il a été compilé et fonctionne sur sa VM correspondante.

3 Assistance à la modélisation et à la vérification formelles

Comme les langages des smart contracts sont formellement définis dans le cadre K, il est possible de procéder à une vérification formelle des smart contracts rédigés dans ces langages. Pour ce faire, il est nécessaire de modéliser formellement leurs exigences, ce qui peut également être fait en utilisant le Cadre K [42].

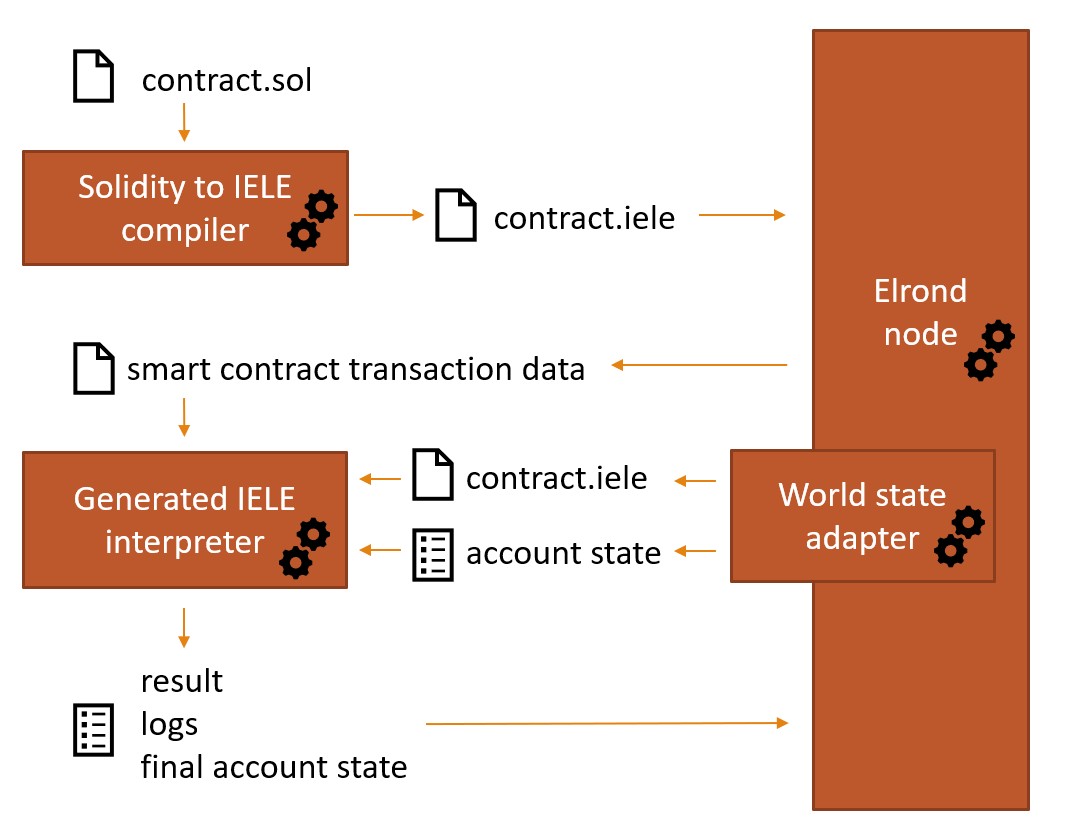


Fig. 8 : Exécution de la Machine Virtuelle Elrond

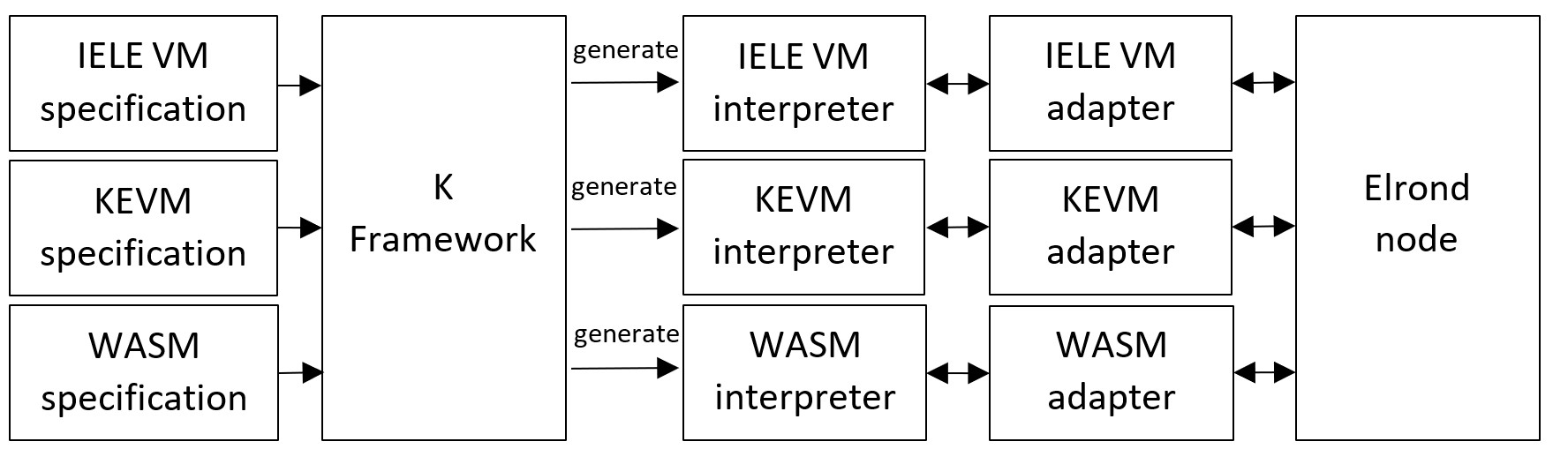


Fig. 9 : Composants de la Machine Virtuelle Elrond

4 Smart Contracts sur une architecture de shards

Les Smart Contracts sur les architectures en Shards n'en sont encore qu'à leurs débuts et posent de sérieux problèmes. Des protocoles comme Atomix [7] ou S-BAC [9] représentent un point de départ. Les dépendances des Smart Contracts dynamiques ne peuvent pas être résolues en déplaçant les Smart Contracts dans le même Shard, car au moment du déploiement, toutes les dépendances ne peuvent pas être calculées.

Solution en cours de recherche dans ce champ :

1) Un mécanisme de verrouillage qui permet l'exécution atomique de Smart Contracts à partir de différents Shards, garantit que les Smart Contracts concernés seront, ou bien tous exécutés en même temps, ou alors aucun d'entre eux. Cela nécessite de multiples messages d'interaction et une synchronisation entre les consensus des différents Shards. [9]

2) La proposition de transfert de contrats inter Shards pour Ethereum 2.0 permettrait de transférer le code et les données de ce Smart Contract dans le Shard de l'appelant au moment de l'exécution. L'exécution atomique n'est pas nécessaire, mais le mécanisme de verrouillage est obligatoire sur le Smart Contract déplacé, ce qui bloquerait l'exécution du Smart Contract pour d'autres transactions. Le mécanisme de verrouillage est plus simple, mais il doit transférer l'ensemble de l'état interne du SC. [43]

Suivant le modèle d'Ethereum, Elrond présente les types de transaction suivants :

1) Construction et déploiement de Smart Contract: l'adresse du destinataire des transactions est vide et le champ de données contient le code du Smart Contract sous forme de tableau d'octets ;

2) Méthode d'invocation du Smart Contract : la transaction contient une adresse de destinataire non vide et cette adresse dispose d'un code associé ;

3) Transactions de paiement : la transaction contient un destinataire non vide et cette adresse ne dispose pas de code associé.

L'approche d'Elrond à ce problème est d'utiliser le modèle d'exécution asynchrone des inter Shards pour le cas de Smart Contracts. L'utilisateur crée une transaction d'exécution de Smart Contract. Si le Smart Contract n'est pas dans le Shard courant, la transaction est traitée comme une opération de paiement, la valeur de la transaction est soustraite du compte émetteur et elle est ajoutée au bloc où réside le Shard émetteur, dans un minibloc avec le Shard de destination où se trouve le compte récepteur. La transaction est notariée par la métachaîne, puis traitée par le Shard de destination. Dans le Shard de destination, la transaction est traitée comme une méthode d'invocation de Smart Contract, car l'adresse du destinataire est un Smart Contract qui existe dans ce tesson. Pour l'appel de Smart Contract, un compte temporaire est créé, en tant qu'image du compte de l'expéditeur, avec le solde de la valeur de la transaction puis le Smart Contract est appelé. Après l'exécution, le Smart Contract peut renvoyer des résultats qui affectent un certain nombre de comptes de différents Shards.

Tous les résultats, qui affectent des comptes à l'intérieur d'un même Shard, sont exécutés au cours du même tour. Pour les comptes qui ne sont pas dans le Shard où le Smart Contract a été exécuté, des transactions appelées "Smart Contract Resultats" seront créées, sauvegardant le résultat de l'exécution du Smart Contract pour chacun de ces comptes. Des miniblocs SCR sont créés pour chaque Shards de destination. Ces miniblocs sont notariés de la même manière que les transactions entre Shards par métachaîne, puis traités par les Shards respectifs, où se trouvent les comptes. Si un Smart Contract appelle dynamiquement un autre Smart Contract à partir d'un autre Shard, cet appel est enregistré comme résultat intermédiaire et traité de la même manière que pour les comptes.

La solution se déroule en plusieurs étapes et la finalisation d'un appel de Smart Contract entre Shards nécessitera au moins 5 tours, mais elle ne requiert pas de verrouillage ni de propagation d'état à travers les Shards.

IX Amorçage et stockage

1 Division du calendrier

Les systèmes à preuve d'enjeu ont tendance à diviser la chronologie en époques et chaque époque en tours plus petits [19]. La chronologie et la terminologie peuvent varier d'une architecture à l'autre, mais la plupart d'entre elles utilisent une approche similaire.

Les époques

3 Dans le protocole Elrond, chaque époque a une durée fixe, initialement fixée à 24 heures (valeur susceptible d'évoluer après quelques étape de confirmation du testnet). Pendant cette période, la configuration des Shards reste inchangée. Le système s'adapte aux exigences de scalabilité entre les époques en modifiant le nombre de Shards. Pour éviter toute collusion, après une époque, la configuration de chaque Shards doit être modifiée. Un remaniement de tous les nœuds entre les Shards permettrait d'obtenir le niveau de sécurité le plus élevé, mais il affecterait la viabilité du système en introduisant une latence supplémentaire due à l'amorçage. C'est pourquoi, à la fin de chaque époque, moins d'un des validateurs éligibles appartenant à un Shard sera redistribué de manière non déterministe et uniforme sur les listes d'attente des autres Shards.

Ce n'est qu'avant le début d'une nouvelle époque que la distribution des validateurs aux Shards peut être déterminée, sans communication supplémentaire, comme le montre la figure 10.

Le processus de remaniement des nœuds se déroule en plusieurs étapes :

1) Les nouveaux nœuds enregistrés à l'époque courante ei atterrissent dans le pool de nœuds non attribués jusqu'à la fin de l'époque actuelle ;

6) Les nœuds remaniés du pool de nœuds attribués sont redistribués avec des ratios plus élevés aux listes d'attente des Shards qui devront se séparer à la prochaine époque ei+2.

Tours

Chaque tour a une durée fixe de 5 secondes (peut subir des mises à jour après plusieurs étapes de confirmation testnet). Lors de chaque tour, un nouveau bloc peut être produit dans chaque partition par un ensemble de validateurs de blocs sélectionnés au hasard (y compris un proposant de blocs). D'un tour à l'autre, l'ensemble est modifié à l'aide de la liste des nœuds éligibles, comme indiqué dans le chapitre IV.

Comme décrit précédemment, la reconfiguration des Shards au sein des époques et la sélection arbitraire des validateurs au sein des cycles découragent la création de coalitions injustes, diminuent la possibilité de DDoS et d'attaques de corruption tout en maintenant la décentralisation et un débit de transactions élevé.

2 Élagage

Un débit élevé conduira à un registre distribué qui croîtra rapidement en taille et augmentera le coût d'amorçage (temps+stockage), comme souligné dans la section XI.1.

Ce coût peut être compensé par l'utilisation d'algorithmes d'élagage efficaces, qui peuvent résumer l'état complet de la BlockChain dans une structure plus condensée. Le mécanisme d'élagage est similaire aux points de contrôle stables de la pBFT [15] et compresse l'ensemble de l'état du livre..

Le protocole Elrond utilise un algorithme d'élagage performant [7] détaillé ci-dessous. Considérons que e est l'époque actuelle et que a est le Shard actuel :

1) les nœuds de shards gardent une trace des soldes des comptes de e dans un arbre Merkle [44] ;

2) à la fin de chaque époque, le proposant de bloc crée un bloc d'état sb(a, e), qui stocke le hachage de la racine de l'arbre Merkle dans l'en-tête du bloc et les soldes dans le corps du bloc ;

3) les validateurs vérifient et exécutent le consensus sur sb(a, e) ;

4) si un consensus est atteint, le proposant du bloc stockera sb(a, e) dans le registre du Shard, ce qui en fera le bloc de genèse pour l'époque e + 1 ;

5) à la fin de l'époque e + 1, les nœuds feront tomber le corps de sb(a, e) et tous les blocs précédant sb(a, e).

Grâce à ce mécanisme, l'amorçage des nouveaux nœuds devrait être très performant. En fait, ils ne partent que du dernier bloc d'état valide et ne calculent que les blocs suivants au lieu de son historique complet.

2) Moins d'un tier (1/3) des nœuds de chaque shard sont sélectionnés aléatoirement pour être remaniés et ajoutés au pool de nœuds attribué ;

Le nouveau nombre de Shards Nsh,i+1 est calculé sur la base du nombre de nœuds dans le réseau ki et de l'utilisation du réseau ;

Les nœuds précédemment présents dans toutes les listes d'attente de Shards, qui sont actuellement synchronisés, sont ajoutés aux listes des validateurs éligibles ;

Les nouveaux nœuds ajoutés à partir du pool de nœuds non attribués sont uniformément répartis de manière aléatoire sur toutes les listes d'attente des Shards pendant l'époque ei+1 ;

X Évaluation de la sécurité

1 Source à caractère aléatoire

Elrond utilise des nombres aléatoires dans son fonctionnement, par exemple pour l'échantillonnage aléatoire des proposants de blocs et des validateurs des groupes de consensus et le mélange des nœuds entre les Shards à la fin d'une époque. Étant donné que ces caractéristiques contribuent aux garanties de sécurité d'Elrond, il est donc important d'utiliser des nombres aléatoires qui sont manifestement imbiaisables et imprévisibles. En plus de ces propriétés, la génération de nombres aléatoires doit également être efficace afin qu'elle puisse être utilisée dans une architecture Blockchain évolutive et à haut débit. Ces propriétés peuvent être trouvées dans certains schémas cryptographiques asymétriques, comme le schéma de signature BLS. Une propriété importante de BLS est que l'utilisation de la même clé privée pour signer le même message produit toujours les mêmes résultats. Ce résultat est similaire à celui obtenu avec l'ECDSA et sa génération déterministe k et il est dû au fait que le système n'utilise aucun paramètre aléatoire :

[ABOVE]

sig = sk · H(m) (8)

où H est une fonction de hashage qui hashe en points sur la courbe utilisée et sk est la clé privée.

2 Création à caractère aléatoire pour Elrond

Un nombre aléatoire est créé dans chaque tour, et ajouté par le promoteur de bloc à chaque bloc de la Blockchain. Cela garantit que les nombres aléatoires sont imprévisibles, car chaque nombre aléatoire est la signature d'un promoteur de bloc différent par rapport à la source aléatoire précédente. La création de nombres aléatoires est décrite ci-dessous dans le cadre d'un cycle de consensus :

1) Le nouveau groupe de consensus est sélectionné à l'aide de la source de caractère aléatoire de l'en-tête de bloc précédent. Le groupe de consensus est formé par un promoteur de bloc et des validateurs.

2) Le proposant du bloc signe la source de caractère aléatoire précédente avec le BLS, ajoute la signature à l'en-tête du bloc proposé comme nouvelle source de caractère aléatoire, puis diffuse ce bloc au groupe de consensus.

3) Chaque membre du groupe de consensus valide la source de caractère aléatoire dans le cadre de la validation du bloc, et envoie sa signature de bloc au promoteur de bloc.

4) Le proposant de bloc agrège les signatures de bloc des validateurs et diffuse le bloc avec la signature de bloc agrégée et la nouvelle source de caractère aléatoire à l'ensemble de la shard.

L'évolution aléatoire de la source dans chaque cycle peut être considérée comme une blockchain non biaisée et vérifiable, où chaque nouveau nombre aléatoire peut être lié et vérifié par rapport au nombre aléatoire précédent.

3 Schéma de finalité du bloc "K"

Le bloc signé au tour n est définitif, si et seulement si les blocs n + 1, n + 2, ..., n + k sont signés. En outre, un bloc final ne peut pas être annulé. La métachaîne ne notifie que les blocs finaux afin de garantir qu'une bifurcation dans une shard n'affecte pas les autres. Celles-ci ne prennent en considération que les blocs finaux de la métachaîne, afin de ne pas être affectés si elle se ramifie. La finalité et l'exactitude sont vérifiées lors de la création et de la validation des blocs. Le paramètre k choisi est 1, ce qui garantit des fourches d'une longueur maximale de 2 blocs. La probabilité qu'une super majorité malveillante (> 2 - n + 1) soit sélectionnée dans le tesson pour le

Le même tour dans le même consensus est de 10-9, même si 33% des nœuds de la shard sont malveillants. Dans ce cas, ils peuvent proposer un bloc et le signer - appelons-le bloc m-, mais il ne sera pas notarié par la métachaîne. La métachaîne ne notifie le bloc m que si le bloc m + 1 est construit par-dessus. Afin de créer le bloc m+1, le groupe de consensus suivant doit être d'accord avec le bloc m. Seul un groupe malveillant sera d'accord avec le bloc m, donc le groupe suivant doit à nouveau avoir une super majorité malveillante. Comme la séquence aléatoire pour la sélection du groupe ne peut être modifiée, la probabilité de sélectionner un autre groupe super majoritaire malveillant est de 10-9 (5,38 - 10-10, pour être exact). La probabilité de signer deux blocs malveillants consécutifs est égale à la sélection de deux sous-groupes comportant au moins (2 - n+1) membres du groupe malveillant en conséquence. La probabilité pour cela est de 10-18. En outre, les groupes sélectionnés en conséquence doivent être de connivence, sinon les blocs ne seront pas signés.

4 Le défi du pêcheur

Lorsqu'un bloc invalide est proposé par une majorité malveillante, la racine de l'état de la shard est altérée avec un résultat invalide (après avoir inclus des modifications invalides à l'arbre d'état). En fournissant la preuve du merkle combiné pour un certain nombre de comptes, un nœud honnête pourrait soulever un défi avec une preuve. Les nœuds honnêtes fourniront le bloc de transactions, l'arbre de merkle réduit précédent avec tous les comptes affectés avant d'appliquer le bloc contesté et les états de contrat intelligents, ce qui permet de démonter la transaction/état invalide. Si une contestation avec la preuve n'est pas fournie dans le délai imparti, le bloc est considéré comme valide. Le coût d'une contestation non valable correspond à la totalité de l'enjeu du nœud qui a soulevé la contestation.

La métachaîne détecte l'incohérence, soit une transaction non valide, soit une racine d'état non valide, grâce aux défis et aux preuves présentés. Elle permet de remonter à la source de l'incohérence et de réduire le groupe de consensus. Dans le même temps, le challenger peut être récompensé par une partie du montant réduit. Un autre problème se pose lorsqu'un groupe malveillant cache le bloc invalide à d'autres nœuds - non malveillants. Cependant, en rendant obligatoire, dans le cadre du consensus actuel, la propagation du bloc produit aux shards soeurs et aux nœuds observateurs, les données ne peuvent plus être cachées. La surcharge de communication est encore réduite en n'envoyant que le minibloc intrasecret aux shards soeurs. Les miniblocs transversaux sont toujours envoyés sur différents sujets accessibles par les nœuds intéressés. En fin de compte, les défis peuvent être relevés par plusieurs nœuds honnêtes. La mise en place de sujets P2P constitue une autre mesure de sécurité. La communication d'une shard vers la métachaîne se fait par un ensemble défini de sujets / canaux, qui peuvent être écoutés par n'importe quel validateur honnête - la métachaîne n'acceptera aucun autre message provenant d'autres canaux. Cette solution introduit un certain retard dans la métachaîne uniquement en cas de sharding, qui sont très peu nombreux et très improbables car, s'ils sont détectés (forte probabilité d'être détectés), les tessons risquent de perdre tout leur enjeu.

5 réorganisation des shards

Après chaque époque, moins de 1 . n des nœuds de chaque shards sont redistribués de manière uniforme et non déterministe sur les autres tessons, afin d'éviter toute collusion. Cette méthode ajoute un surcoût d'amorçage pour les nœuds qui ont été redistribués, mais n'affecte pas la qualité de vie car les nœuds remaniés ne participent pas au consensus de l'époque à laquelle ils ont été redistribués. Le mécanisme d'élagage diminuera cette fois-ci jusqu'à un montant réalisable, comme expliqué dans la section IX.2.

6 Sélection du groupe de consensus

Après chaque tour, un nouvel ensemble de validateurs est sélectionné en utilisant la graine aléatoire du dernier bloc validé, le tour en cours et la liste des nœuds éligibles. En cas de désynchronisation du réseau due à des retards dans la propagation des messages, le protocole dispose d'un mécanisme de récupération, et prend en considération à la fois le cycle r et la semence aléatoire du dernier bloc validé afin de sélectionner de nouveaux groupes de consensus à chaque cycle. Cela évite les bifurcations et permet la synchronisation sur le dernier bloc. La petite fenêtre de temps (temps de ronde) dans laquelle les validateurs

est connu, minimise les vecteurs d'attaque.

7 Evaluation des nœuds

Outre l'enjeu, la notation du validateur éligible influence les chances d'être sélectionné dans le cadre du groupe de consensus. Si le ppromoteur de bloc est honnête et que son bloc s'engage dans la Blockchain, sa note sera augmentée, sinon, sa note sera diminuée. De cette façon, chaque validateur possible est incité à être honnête, à utiliser la version la plus récente du logiciel client, à augmenter sa disponibilité de service et à assurer ainsi que le réseau fonctionne comme prévu.

8 La redondance des fragments (shards)

Les nœuds qui ont été distribués en tessons de frères et sœurs au niveau le plus bas de l'arbre (voir section IV.4) gardent la trace des données de la Blockchain et de l'état de l'application de chacun. En introduisant le concept de redondance des shards, lorsque le nombre de nœuds dans le réseau diminue, certaines des shards soeurs devront être fusionnés. Les nœuds ciblés lanceront instantanément le processus de fusion des shards.

XI Comprendre les vrais problèmes

1 Centralisé vs Décentralisé

La blockchain a été initialement instanciée comme alternative au système financier centralisé des systèmes [45]. Même si la liberté et l'anonymat des architectures distribuées restent un avantage incontesté, les performances doivent être analysées à l'échelle mondiale dans un environnement réel.

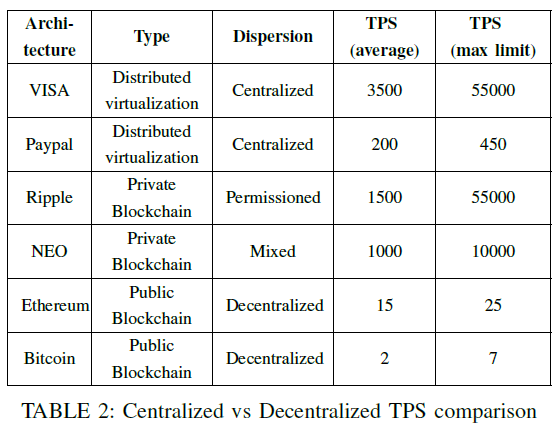
La mesure la plus pertinente de la performance est le nombre de transactions par seconde (TPS), comme le montre le tableau 2. Une comparaison des transactions par seconde entre les systèmes centralisés traditionnels et les nouvelles architectures décentralisées, dont la fiabilité et l'efficacité ont été validées à grande échelle, reflète une réalité objective mais troublante [46], [47], [48], [49]

[ABOVE]

La scalabilité des architectures en BlockChains est un problème critique mais toujours non résolu. Prenons, par exemple, l'exemple de la détermination des implications en matière de stockage des données et de bootstrapping des architectures de blockchain actuelles qui se mettraient soudainement à fonctionner au même niveau de débit que Visa. En procédant à de tels exercices, l'ampleur des multiples problèmes secondaires devient évidente (voir la figure 11).

<Voir 287>

<voir 287>



XII Le paradigme de performance blockchain

Le processus de conception d'architectures distribuées sur Blockchain fait face à plusieurs défis, peut-être l'un des plus difficiles étant la lutte pour maintenir l'opérabilité dans des conditions de pression propres au contexte. Les principaux composants qui déterminent la pression de performance sont :

• complexité

• taille du système

• volume des transactions

Complexité

Le premier élément qui limite les performances du système, est le protocole de consensus. Un protocole plus compliqué détermine un point critique plus gros. Dans les architectures consensuelles de PoW, une grande pénalité de performance est induite par la complexité de l'exploitation qui vise à maintenir le système décentralisé et la résilience des ASIC [50]. Pour surmonter ce problème, le PoS fait un compromis, simplifie la gestion du réseau en concentrant la puissance de calcul sur un sous-ensemble du réseau, mais apporte plus de complexité au mécanisme de contrôle.

Architecture

Type Dispersion

Distribué

TPS (moyenne)

TPS (limite maximale)

Taille du système

L'augmentation du nombre de nœuds dans les architectures validées existantes entraîne une grave dégradation des performances et induit

VISA

Paypal Ripple NEO Ethereum Bitcoin

virtualisation

Virtualisation distribuée Blockchain privée Blockchain privé publique Blockchain publique

Blockchain

Centralisé 3500 55000

Centralisé 200 450

Avec permission 1500 55000

Mixte 1000 10000

Décentralisé 15 25

Décentralisé 2 7

un prix de calcul plus élevé qui doit être payé. Les Shards semblent être une bonne approche, mais la taille des Shards joue un rôle majeur. Les Shards plus petits sont agiles mais plus susceptibles d'être affectés par des groupes malveillants, les Shards plus gros sont plus sûrs, mais leur reconfiguration affecte la viabilité du système.

Transaction volume

Plus pertinent que les autres, le dernier élément de la liste concerne les performances de traitement des transactions. Afin de mesurer correctement l'impact de ce critère, celui-ci doit être analysé selon les deux points de vue suivants :

TABLEAU 2 : Comparaison des TPS entre une architecture centralisée et décentralisée

C1 Débit des transactions - combien de transactions un système peut traiter par unité de temps, connue sous le nom de TPS, en sortie d'un système donné [51] ;

- C2 Finalité de la transaction - la vitesse à laquelle une action particulière est traitée, en se référant à l'intervalle entre son lancement et sa finalisation - depuis l'entrée vers la sortie.

C1. Le débit de transactions dans les architectures à chaîne unique est très faible et peut être augmenté en utilisant des solutions de contournement telles que la chaîne latérale [52]. Dans une architecture en Shards comme la nôtre, le débit de transaction est influencé par le nombre de Shards, les capacités de calcul des validateurs/proposeurs de blocs et l'infrastructure de messagerie [8]. En général, comme le montre la figure 13, cela va bien au public, mais malgré l'importance de la métrique, cela ne donne qu'une vue fragmentée. C2. Finalité de la transaction - Un aspect plus délicat qui souligne que même si le système peut avoir un débit de 1000 TPS, il peut prendre un certain temps pour traiter une transaction particulière. Outre les capacités de calcul des validateurs/proposeurs de bloc et l'infrastructure de messagerie, la finalité de la transaction est principalement affectée par l'algorithme de répartition (lorsque la décision est prise) et le protocole de routage (où la transaction doit-elle être exécutée). La plupart des architectures de pointe existantes refusent de mentionner cet aspect, mais du point de vue de l'utilisateur, il est extrêmement important. C'est ce qu'illustre la figure 14, où le temps total nécessaire pour exécuter une certaine transaction du début à la fin est pris en compte.

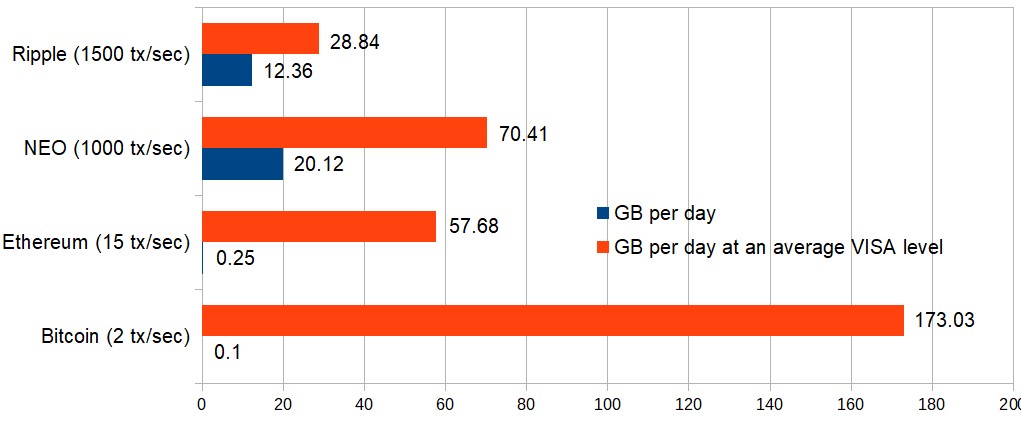


Fig. 11 : Estimation du stockage - Architectures distribuées validées pour un fonctionnant à une moyenne du TPS VISA

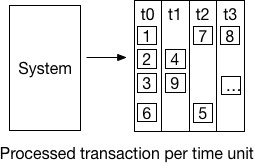


Fig. 13 : Débit des transactions

<cf 321>

À Elrond, le mécanisme de répartition (détaillé dans la section V) permet d'améliorer le délai de traitement en acheminant les transactions directement vers le bon tesson, ce qui réduit les délais globaux.

XIII Conclusion

1 Performance

Les tests et simulations de performance, présentés à la figure 12, reflètent l'efficacité de la solution en tant que registre distribué hautement scalable. Comme de plus en plus de nœuds rejoignent le réseau, notre approche de Sharding démontre une augmentation linéaire du débit. Le modèle de consensus choisi implique de multiples tours de communication, le résultat est donc fortement influencé par la qualité du réseau (vitesse, latence, disponibilité). Les simulations réalisées avec notre testnet en utilisant les moyennes de vitesse du réseau mondial, à sa limite théorique maximale, suggèrent qu'Elrond dépasse le niveau VISA moyen avec seulement 2 Shards, et se rapproche du niveau VISA maximum avec 16 Shards.

Fig. 14 : Finalité de la transaction

Recherche actuelle et future

Notre équipe réévalue et améliore constamment la conception d'Elrond, afin d'en faire l'une des architectures de BlockChain publique des plus prometteuses ; elle résout le problème de la sclabilité par le Sharding d'état adaptatif, tout en maintenant la sécurité et une haute efficacité énergétique grâce à un mécanisme sécurisé de consensus de preuve d'enjeux. Voici quelques-unes de nos prochaines orientations en matière d'amélioration :

1) Renforcement de l'apprentissage : nous visons à accroître l'efficacité du processus de Sharding en répartissant les clients qui négocient fréquemment dans le même Shard afin de réduire le coût global ;

2) IA pour la supervision : créer un superviseur en IA qui détecte les modèles de comportement malveillant ; on ne sait pas encore comment cette fonctionnalité peut être intégrée dans le protocole sans perturber la décentralisation ;

3) Fiabilité en tant que facteur de consensus : le protocole existant pondère entre l'enjeu (staking) et la notation (rating), mais nous prévoyons d'ajouter la fiabilité, comme une mesure qui devrait être calculée de manière distribuée après l'application d'un protocole de consensus sur les blocs précédemment soumis récemment.

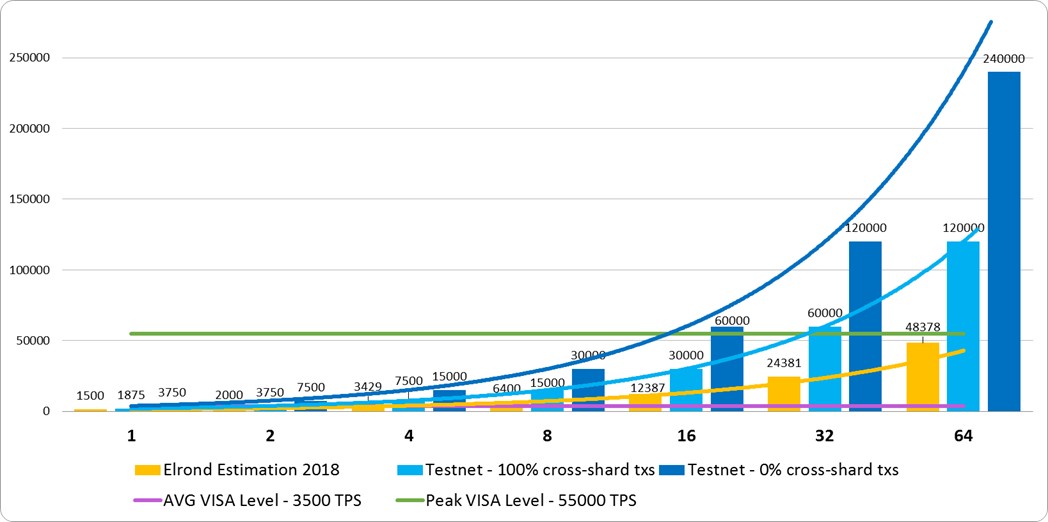


Fig. 12 : Débit réseau mesuré en transactions par seconde avec une vitesse globale sur le réseau de 8 Mo/s

l'histoire;

4) interopérabilité inter-blockchain : mettre en œuvre et rendre hommage à des normes telles que celles lancées par la Decentralized Identity Foundation [53] ou la Blockchain Interoperability Alliance [54] ;

5) Transactions préservant la vie privée : utiliser le "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge" [55] pour protéger l'identité des participants et offrir des capacités d'audit tout en préservant la vie privée.

3 Conclusions générales

Elrond est la première BlockChain publique hautement scalable qui utilise l'algorithme récemment proposé de Preuve d'Enjeu Sécurisée (SPoS) dans une véritable architecture de Sharding d'état pour atteindre un débit équivalent à celui de VISA et des temps de confirmation de quelques secondes. L'approche novatrice d'Elrond sur le Sharding d'états adaptatif améliore la proposition d'Omniledger en augmentant la sécurité et le débit, tandis que le routage automatique des transactions et les mécanismes intégrés de redondance d'états réduisent considérablement les temps de latence. En utilisant une technique de suppression des Shards, les coûts d'amorçage et de stockage sont également considérablement réduits par rapport aux autres approches. L'algorithme de consensus de Preuve d'Enjeu Sécurisée (SPoS) récemment introduit garantit une équité distribuée et améliore l'idée de sélection aléatoire d'Algorand, en réduisant le temps nécessaire pour la sélection aléatoire du groupe de consensus de 12 secondes à 100 ms. Notre méthode de combinaison du Sharding d'états et du très efficace algorithme de consensus de Preuve d'Enjeu Sécurisée (SPoS) a donné des résultats prometteurs dans nos estimations initiales, validées par les derniers résultats de notre testnet.

References

[1] G. Hileman and M. Rauchs, “2017 Global Cryptocurrency Benchmarking Study,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2965436, Apr. 2017. [Online]. Available: https://papers.ssrn.com/abstract=2965436

[2] “The Ethereum Wiki - Sharding FAQ,” 2018, original-date: 2014-02-

14T23:05:17Z. [Online]. Available: https://github.com/ethereum/wiki/

wiki/Sharding-FAQ

[3] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, “Algorand: Scaling Byzantine Agreements for Cryptocurrencies,” in Proceedings of the 26th Symposium on Operating Systems Principles, ser. SOSP ’17. New York, NY, USA: ACM, 2017, pp. 51–68. [Online]. Available: http://doi.acm.org/10.1145/3132747.3132757

[4] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in Advances in Cryptology – ASIACRYPT ’01, LNCS. Springer,

2001, pp. 514–532.

[5] D. Boneh, M. Drijvers, and G. Neven, “Compact multi-signatures for smaller blockchains,” in Advances in Cryptology – ASIACRYPT 2018, ser. Lecture Notes in Computer Science, vol. 11273. Springer, 2018, pp. 435–464.

[6] V. Buterin, “Ethereum: A Next-Generation Smart Contract and

Decentralized Application Platform,” 2013. [Online]. Available: https:

//www.ethereum.org/pdfs/EthereumWhitePaper.pdf

[7] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, “OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding,” Tech. Rep. 406, 2017. [Online]. Available: https://eprint.iacr.org/2017/406

[8] “The ZILLIQA Technical Whitepaper,” 2017. [Online]. Available:

https://docs.zilliqa.com/whitepaper.pdf

[9] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, “Chainspace: A Sharded Smart Contracts Platform,” arXiv:1708.03778 [cs], Aug. 2017, arXiv: 1708.03778. [Online]. Available: http:

//arxiv.org/abs/1708.03778

[10] G. Wood, “Ethereum: A Secure Decentralised Generalised Transaction Ledger,” 2017. [Online]. Available: https://ethereum. github.io/yellowpaper/paper.pdf

[11] “Solidity — Solidity 0.4.21 documentation.” [Online]. Available:

https://solidity.readthedocs.io/en/v0.4.21/

[12] “web3j,” 2018. [Online]. Available: https://github.com/web3j

[13] “Casper,” 2018. [Online]. Available: http://ethresear.ch/c/casper

[14] “The State of Ethereum Scaling, March 2018 – Highlights from EthCC on Plasma Cash, Minimum Viable Plasma, and More. . . – Medium,” 2018. [Online]. Available: https://medium.com/loom-network/ the-state-of-ethereum-scaling-march-2018-74ac08198a36

[15] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” in Proceedings of the Third Symposium on Operating Systems Design and Implementation, ser. OSDI ’99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186. [Online]. Available: http://dl.acm.org/citation.cfm?id=296806.296824

[16] Y. Jia, “Op Ed: The Many Faces of Sharding for Blockchain Scalability,” 2018. [Online]. Available: https://bitcoinmagazine.com/ articles/op-ed-many-faces-sharding-blockchain-scalability/

[17] “Using Merklix tree to shard block validation | Deadalnix’s den,” 2016. [Online]. Available: https://www.deadalnix.me/2016/11/06/ using-merklix-tree-to-shard-block-validation/

[18] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” p. 9,

2008.

[19] “Why we are building Cardano - Introduction.” [Online]. Available:

https://whycardano.com/

[20] “Constellation - a blockchain microservice operating system - White Paper,” 2017, original-date: 2018-01-05T20:42:05Z. [Online]. Available: https://github.com/Constellation-Labs/Whitepaper

[21] “Bitshares - Delegated Proof-of-Stake Consensus,”

2014. [Online]. Available: https://bitshares.org/technology/

delegated-proof-of-stake-consensus/

[22] dantheman, “DPOS Consensus Algorithm - The Missing White Paper,” May 2017. [Online]. Available: https://steemit.com/dpos/@dantheman/ dpos-consensus-algorithm-this-missing-white-paper

[23] “EOS.IO Technical White Paper v2,” 2018, original-date: 2017-

06-06T07:55:17Z. [Online]. Available: https://github.com/EOSIO/ Documentation/blob/master/TechnicalWhitePaper.md

[24] C. Paar and J. Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners. Berlin Heidelberg: Springer-Verlag, 2010. [Online]. Available: //www.springer.com/gp/book/9783642041006

[25] C. Schnorr, “Efficient signature generation by smart cards,” Journal of

Cryptology, vol. 4, pp. 161–174, Jan. 1991.

[26] K. Michaelis, C. Meyer, and J. Schwenk, “Randomly Failed!

The State of Randomness in Current Java Implementations,” in Topics in Cryptology – CT-RSA 2013, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Feb. 2013, pp.

129–144. [Online]. Available: https://link.springer.com/chapter/10.1007/

978-3-642-36095-4 9

[27] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, “Twisted

Edwards Curves,” in Progress in Cryptology – AFRICACRYPT

2008, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Jun. 2008, pp. 389–405. [Online]. Available: https:

//link.springer.com/chapter/10.1007/978-3-540-68164-9 26

[28] A. Poelstra, “Schnorr Signatures are Non-Malleable in the Random Oracle Model,” 2014. [Online]. Available: https://download.wpsoftware. net/bitcoin/wizardry/schnorr-mall.pdf

[29] C. Decker and R. Wattenhofer, “Bitcoin Transaction Malleability and

MtGox,” arXiv:1403.6676 [cs], vol. 8713, pp. 313–326, 2014, arXiv:

1403.6676. [Online]. Available: http://arxiv.org/abs/1403.6676

[30] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, “Simple Schnorr Multi-Signatures with Applications to Bitcoin,” Tech. Rep. 068, 2018. [Online]. Available: https://eprint.iacr.org/2018/068

[31] Y. Seurin, “On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model,” in Advances in Cryptology – EUROCRYPT 2012, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Apr. 2012, pp. 554–571. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-29011-4 33

[32] K. Itakura and K. Nakamura, “A public-key cryptosystem suitable for digital multisignatures,” 1983.

[33] S. Micali, K. Ohta, and L. Reyzin, “Accountable-subgroup Multisignatures: Extended Abstract,” in Proceedings of the 8th ACM Conference on Computer and Communications Security, ser. CCS ’01. New York, NY, USA: ACM, 2001, pp. 245–254. [Online]. Available: http://doi.acm.org/10.1145/501983.502017

[34] T. Ristenpart and S. Yilek, “The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks,” in Advances in Cryptology - EUROCRYPT 2007, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, May 2007, pp.

228–245. [Online]. Available: https://link.springer.com/chapter/10.1007/

978-3-540-72540-4 13

[35] M. Bellare and G. Neven, “Multi-signatures in the Plain public-Key Model and a General Forking Lemma,” in Proceedings of the 13th ACM Conference on Computer and Communications Security, ser. CCS ’06. New York, NY, USA: ACM, 2006, pp. 390–399. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180453

[36] D.-P. Le, A. Bonnecaze, and A. Gabillon, “Multisignatures as Secure as the Diffie-Hellman Problem in the Plain Public-Key Model,” in Pairing-Based Cryptography – Pairing 2009, ser. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, Aug. 2009, pp.

35–51. [Online]. Available: https://link.springer.com/chapter/10.1007/

978-3-642-03298-1 3

[37] T. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen, “Adding Concurrency to Smart Contracts,” in Proceedings of the ACM Symposium on Principles of Distributed Computing, ser. PODC ’17. New York, NY, USA: ACM, 2017, pp. 303–312. [Online]. Available: http://doi.acm.org/10.1145/3087801.3087835

[38] J. Kwon and E. Buchman, “Cosmos Network - Internet of Blockchains,”

2017. [Online]. Available: https://cosmos.network/whitepaper

[39] G. Ros, u and T. F. S, erba˘nuta˘, “An overview of the k semantic frame- work,” The Journal of Logic and Algebraic Programming, vol. 79, no. 6, pp. 397–434, 2010.

[40] T. Kasampalis, D. Guth, B. Moore, T. Serbanuta, V. Serbanuta, D. Fi- laretti, G. Rosu, and R. Johnson, “Iele: An intermediate-level blockchain language designed and implemented using formal semantics,” Tech. Rep., 2018.

[41] E. Hildenbrandt, M. Saxena, X. Zhu, N. Rodrigues, P. Daian, D. Guth, and G. Rosu, “Kevm: A complete semantics of the ethereum virtual machine,” Tech. Rep., 2017.

[42] “How Formal Verification of Smart Contracts Works | RV Blog.” [Online]. Available: https://runtimeverification.com/blog/ how-formal-verification-of-smart-contracts-works/

[43] “Cross-shard contract yanking.” [Online]. Available: https://ethresear. ch/t/cross-shard-contract-yanking/1450

[44] R. C. Merkle, “A Certified Digital Signature,” in Advances in Cryptology

— CRYPTO’ 89 Proceedings, ser. Lecture Notes in Computer Science. Springer, New York, NY, Aug. 1989, pp. 218–238. [Online]. Available: https://link.springer.com/chapter/10.1007/0-387-34805-0 21

[45] A. Veysov and M. Stolbov, “Financial System Classification: From Conventional Dichotomy to a More Modern View,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2114842, Jul. 2012. [Online]. Available: https://papers.ssrn.com/abstract=2114842

[46] “XRP - The Digital Asset for Payments.” [Online]. Available:

https://ripple.com/xrp/

[47] “Visa - Annual Report 2017,” 2018. [Online]. Avail- able: https://s1.q4cdn.com/050606653/files/doc financials/annual/2017/ Visa-2017-Annual-Report.pdf

[48] “PayPal Reports Fourth Quarter and Full Year 2017 Results (NASDAQ:PYPL),” 2018. [Online]. Available: https://investor. paypal-corp.com/releasedetail.cfm?releaseid=1055924

[49] M. Schwarz, “Crypto Transaction Speeds 2018 - All the Major Cryptocurrencies,” 2018. [Online]. Available: https://www.abitgreedy. com/transaction-speed/

[50] “The Ethereum Wiki - Mining,” 2018, original-date: 2014-02-

14T23:05:17Z. [Online]. Available: https://github.com/ethereum/wiki/

wiki/Mininghttps://github.com/ethereum/wiki

[51] “Transaction throughput.” [Online]. Available: https://docs.oracle.com/

cd/E17276 01/html/programmer reference/transapp throughput.html

[52] W. Martino, M. Quaintance, and S. Popejoy, “Chainweb: A Proof- of-Work Parallel-Chain Architecture for Massive Throughput,” 2018. [Online]. Available: http://kadena.io/docs/chainweb-v15.pd

[53] “DIF - Decentralized Identity Foundation.” [Online]. Available:

http://identity.foundation/

[54] H. I. World, “Blockchain Interoperability Alliance: ICON x Aion x Wanchain,” Dec. 2017. [Online]. Available: https://medium.com/helloiconworld/ blockchain-interoperability-alliance-icon-x-aion-x-wanchain-8aeaafb3ebdd

[55] S. Goldwasser, S. Micali, and C. Rackoff, “The Knowledge Complexity of Interactive Proof-systems,” in Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, ser. STOC ’85. New York, NY, USA: ACM, 1985, pp. 291–304. [Online]. Available: http://doi.acm.org/10.1145/22145.22178