Constellation

Um sistema para operação de micro serviços em Blockchain

25 de Novembro 2017

Resumo

As atuais tecnologias blockchain, como Bitcoin, Ethereum, e outras, possuem limitações síncronas sistêmicas, que as tornam inviáveis para a adoção em massa na atual arquitetura das aplicações descentralizadas voltadas aos consumidores. Os mecanismos de consenso e contratos inteligentes não conseguem escalar a um nível adequado para atender à classe consumidora de forma funcional. Nossa proposta é um sistema distribuído resistente à falhas, horizontalmente escalável, que pode implementar full nodes em um cliente mobile. Portanto, nós apresentamos uma reformulação do consenso criptográfico seguro, em uma arquitetura server-less, a qual utiliza uma ExtendedTrustChain assíncrona, consenso de Proof of Meme, contratos inteligentes que permitem a composição de micro serviços, utilizando o JVM e inicialmente implementados como uma máquina de estado finitos baseados no ator. Essa arquitetura garante um alto volume de transações, permitindo, dessa forma, que aplicações distribuídas voltadas para o mercado consumidor sejam construídas na Constellation.

Introdução

Embora o grande objetivo de descentralização, criptomoedas são operadas e asseguradas cada vez mais por organizações de redes e mineradoras centralizadas, que controlam a segurança do sistema[1]. Com o crescimento de redes ''pós-Bitcoin'' como a Ethereum, a meta inicial de apenas processar operações financeiras expandiu-se para o fornecimento de uma computação distribuída, por meio do EVM e seu sistema de lógica, com a confiança de não depender de nenhum terceiro. Enquanto implementações criativas e diversas desses protocolos existem, elas são totalmente isoladas dos softwares mainstream existentes, e, decorrente disso, há uma grande barreira para acessá-las[2].Além de tomar muito tempo, é de alto custo o acesso para empreendimentos desenvolverem, implantarem e manterem sistemas de aplicações distribuídas robustas, que utilizam linguagens de programação e padrões de design pouco

[1] Digital Currency Group 'Bitcoin Scaling Agreement at Consensus 2017' medium.com/@DCGco
[2] J. Evans 'Blockchains are the new Linux, not the new internet' techcrunch.com

familiares. Além disso, escalar tais aplicações para a demanda de performance até mesmo de uma necessidade de um empreendimento modesto, é praticamente impossível no atual cenário de blockchains públicas. Para superar esses desafios técnicos, a tecnologia blockchain requer um protocolo auto-sustentável que funcione como um sistema operacional distribuído horizontalmente escalável, econômico e eficiente.

Bitcoin foi criado para resolver o problema do consenso distribuído para transações financeiras, mas contou com o processo de consenso intensivo de energia conhecido como prova de trabalho. Isso agrupa o poder computacional vertical de um grupo seleto de indivíduos com poder de computação bem desenvolvido e, portanto, centralizado, o que coloca a segurança da rede e as recompensas nas mãos dos mesmos poucos selecionados. À medida que os debates sobre escalabilidade aumentaram dentro dessas conspirações, isso levou à criação de vários forks do Bitcoin, como o Bitcoin Cash e o Bitcoin Gold. Essa separação em duas 'facções' é extremamente prejudicial ao ecossistema. Requisitos de taxa de transferência antiquada (ou seja, tamanho de bloco limitado a 1 MB) combinados com um algoritmo que utiliza de forma intensiva o tempo fizeram com que os tempos de transação e as taxas de transação disparassem. O Ethereum usa um algoritmo de consenso de prova de trabalho semelhante ao ASIC e está migrando para um mecanismo de consenso de prova de participação, o Casper. Com a comprovação de que há um desequilíbrio democrático, aqueles com mais dinheiro conseguem escolher o estado da rede por meio de consenso. A Ethereum foi pioneira no uso de contratos inteligentes, no entanto, devido à sua execução síncrona, aplicativos distribuídos prontos para o consumidor são severamente limitados. A concorrência é a chave para fornecer produtos focados no consumidor e a Constellation obtém isso com seu contrato inteligente como arquitetura de microsserviço.

O que torna a Constellation diferente?

A comunidade de desenvolvimento de blockchain tem procurado novas implementações de tecnologia de contabilidade distribuída para resolver o problema de escalabilidade e até agora não conseguiu resolver isso. O problema ainda permanece: como podemos processar mais transações e executar mais serviços em menos tempo com menos custo? Propomos uma abordagem de escala horizontal que aplica técnicas semelhantes ao MapReduce.

Escalabilidade horizontal é a aplicação da programação concorrente; Isso significa que, conforme os usuários ingressam na rede, a taxa de transferência da transação aumenta. O MapReduce é um processo de decomposição de cálculos em operações simples que podem ser inseridas em um DAG assíncrono (gráfico acíclico direcionado) de cálculo, aumentando assim a eficiência de um programa já concorrente.

O protocolo Constellation implementa uma arquitetura blockchain horizontalmente escalável conhecida como Cadeia de Confiança Estendida(Extended Trust Chain) com uma camada ponto a ponto conhecida como “protocolo de fofoca”, que pode ser implantada em um dispositivo móvel. O Constellation aborda contratos inteligentes com uma arquitetura de microsserviço, permitindo que os serviços altamente disponíveis sejam encadeados e compostos em aplicativos distribuídos com apenas um entendimento do SLA (contrato de nível de serviço) de cada microsserviço e / ou assinatura de tipo.

O “protocolo de fofoca” permite que grandes redes comuniquem o estado total da rede em uma escala que é maior do que a tecnologia blockchain existente. Nesse tipo de rede, cada membro da rede monitora seus vizinhos e, quando recebe novas mensagens, propaga essa mensagem para todos os seus vizinhos. Isso tem algumas propriedades matemáticas interessantes, mas no nosso caso, permite que uma grande gama de dispositivos conectados compartilhem o estado da rede e consigam um consenso em uma escala inédita nas implementações atuais de blockchain (veja a Figura 1).

Figura

Uma abordagem ao consenso de blockchain que permite escalabilidade horizontal é a ExtendedTrustChain. Na ExtendedTrustChain, existem vários tipos de nós, cada um com sua própria responsabilidade e função na rede para controlar de maneira assíncrona diferentes aspectos do protocolo.

Figura 2

Nós

Especificamente, existem nós básicos que enviam transações e hospedam a cadeia individual de um membro (a cadeia de cada indivíduo forma um DAG com outras pessoas na rede quando uma transação é feita e é dividida em blocos lineares).

Processos de Checkpoint

Existem processos de nó que realizam consenso nas transações (que preenchem um buffer e são codificados em blocos de pontos de verificação). Essas coleções de transações são fragmentadas e se tornam o próximo bloco na cadeia principal.

Processos do validador

Finalmente, há nós validadores que executam processos de validação e hospedam o estado atual da cadeia e, portanto, distribuem o estado da cadeia ou histórico para os nós na rede.

EVM vs JVM

A formulação de aplicações distribuídas como operações MapReduce em contratos inteligentes foi proposta pela atualização do Plasma para a Ethereum. No entanto, com sua aplicação no EVM, ele ainda sofre as mesmas armadilhas da arquitetura síncrona para o dimensionamento de uma maneira econômica e de baixa latência. A linguagem de programação Solidity foi construída para abordar os riscos do não-determinismo no desenvolvimento de contratos inteligentes. A necessidade de uma instrução determinística se origina da necessidade de calcular o custo da execução de um contrato Inteligente e como preventivo de medida contra ataques DDOS e spam de mensagens para a rede. Contudo, se o código compilado puder ser registrado em uma cadeia e verificado em uma JVM, então a compilação determinista torna-se um requisito mais flexível. Portanto, a redução no consumo de gás e contornar os limites dos contratos inteligentes , com a complexidade de micro serviços, se tornaria uma possibilidade.

É intuitivo projetar aplicativos distribuídos como composições de blocos de construção, como montar Legos, onde cada peça é um micro serviço existente. Dando um passo além, implementamos a interoperabilidade em cadeia que permite que as empresas de blockchain forneçam aplicativos distribuídos para reutilização por outros como blocos de construção para aplicativos distribuídos mais complexos. Um verdadeiro ecossistema descentralizado não pode existir com contratos inteligentes monolíticos, tornando a reutilização deste código impossível e ineficiente.

O uso da JVM como um canal para aplicativos distribuídos é uma indústria padrão na comunidade de Big Data. Aplicacões para consumidores são implantadas como uma coleção de microsserviços em grupos de escalonamento automático que dependem de fácil suprimento, o que a JVM fornece. Nestes cenários, mil grupos de nós podem ser abastecidos e dimensionados automaticamente em instâncias virtuais da AWS. A mesma escalabilidade pode ser realizada com a arquitetura de microsserviços da Constellation, que permite o provisionamento de recursos conforme necessário; fornecendo assim uma dinamicamente escalável arquitetura de aplicativos distribuídos.

Esse tipo de arquitetura distribuída requer inerentemente a modularidade que a programação funcional fornece. Portanto, estamos usando a linguagem de programação Scala para construir uma interface que pode ser implementada por qualquer linguagem JVM. Quando micro serviços são projetados como tipos concretos, aplicações distribuídas podem ser compostas deles diretamente no código-fonte, permitindo a validação em tempo de compilação. Os

desenvolvedores podem criar novas funcionalidades encadeando também contratos inteligente aplicáveis à funcionalidades compostas; uma constelação de lógica computacional é, portanto, composta por um ou mais micro serviços ou distribuídas aplicações. A interface de contrato inteligente do Constellation será projetada com digitação covariante e contravariante em mente, o que significa que se misturarmos micro serviços incompatíveis, saberemos imediatamente. Por causa disso, podemos até criar constelações incluindo contratos inteligentes de outros blockchains, apresentando soluções para problemas de cross-chain como o Polkadot.

Considerando o nosso ethos funcional de programação, vale notar que o nosso protocolo de consenso poderia ser descrito pela definição teórica da categoria de hilomorfismo.

HyloChain - Arquitetura de consenso

Nós nomeamos nossa arquitetura de consenso HyloChain por utilizarmos a arquitetura de consenso hilomórfica. Uma rodada de consenso coleta o hash do bloco resultante de uma rodada anterior, e adiciona-o como uma transação regular para a piscina de transações. O preenchimento dessa piscina é isomorfo para uma operação de desdobramento. Uma vez que este bloco de verificação é preenchido com a rodada anterior, com mais novas transações, ele é assinado. Ou seja, é isomorfo a uma operação de dobra.

Definição: HyloChain

Um hilomorfismo h: A → C pode ser definido em termos de seu anamórfico separado e por partes catamórficas. Tome como definição de um hilomorfismo:

h = [(c, ⊕), (g, p)] (1)

A parte anamórfica pode ser definida em termos de uma função unária g: A → BA definindo a lista de elementos em B através da aplicação repetida ou desdobramento, e um predicado p: A → Boolean fornecendo a condição de terminação.

A parte catamórfica pode ser definida como uma combinação de um valor inicial c ∈ C para a dobra e um operador binário ⊕: B × C → C usado para executar uma dobra.

7G. Wood et al. https://github.com/w3f/polkadot-whitepaper/raw/master/PolkaDotPaper.pdf

No nosso caso, a 'fofoca' das mensagens é o nosso anamorfismo e o hashing dessas mensagens mais o resultado do bloco anterior é o nosso catamorfismo.

Se um operador for definido como uma função criptográfica de hash, ⊕ = CHash e g n = (n, n - 1) e p n = Falso (como nossa cadeia não tem condição de terminação) para a enésima iteração, o HyloChain é definido categoricamente como

HyloChain = [(GenesisBlock, ⊕), (g, p)] (2)

Onde o GenesisBlock é o nosso elemento inicial, ou seja, o bloco de gênese usado quando implementamos a HyloChain.

Relação com a Cadeia de Confiança Extendida (ExtendedTrustChain)

Nossa arquitetura de consenso, o HyloChain se baseia na arquitetura ExtendedTrustChain e se expande das seguintes maneiras.Nessa cadeia, cada nó opera como uma conta, mantém o histórico de sua própria cadeia, e depende da ordem de transação para validadação (semelhante ao HashGraph8). As transações são assinadas pelo iniciador e contraparte, sendo logo depois transmitidas para a rede (via fofoca). Delegados realizam consenso sobre as transações dentro de um bloco de checkpoint, e são escolhidos com base em sua reputação. O resultado do consenso é transmitido novamente e adicionado como uma transação normal para o próximo bloco. 9

Tolerância ao erro

A tolerância a falhas do HyloChain da Constellation segue um típico modelo de consenso bizantino. A probabilidade de um adversário malicioso assumir o controle do consenso da rede se aproxima de 1/2 à medida que o controle dos participantes do consenso se aproxima de 1/310.

Para garantir a segurança da rede, podemos usar esses limites para escolher parâmetros para ajustar a tolerância a falhas enquanto maximiza a frequência de transferências

10M. Pease, R. Shostak and L. Lamport, 'Reaching agreement in the presence of faults', Journal of the ACM (JACM), vol. 27, no. 2

Figura 4: Probabilidade de selecionar mais que (n − 1) / 3 (denotado como bolas pretas, visto que isso vem de uma explicação de uma distribuição hipergeométrica) para diferentes valores de t (onde t é o número de atores bizantinos) com um tamanho de população fixado à 2000. K. Cong et al.

Além disso, com base em nosso procedimento de consenso bizantino (estamos tentando com a utilização de ACS, como implementado por M. Ben-Or et al. que é usado por Honeybadger BFT) e a dupla assinatura de transações, é impossível bifurcar a rede. O único resultado de um consenso incorreto seria a censura individual das transações de um bloco de ponto de verificação, ou seja, uma transação perdida durante o consenso. Isso não causaria perda de capital e a mitigação é tão simples quanto reenviar para a rede.

A relação de rendimento é linear em relação ao número de participantes dada a capacidade de escala horizontal, mas um aumento linear na tolerância a falhas custa um aumento polinomial no custo de comunicação11. Assim, devemos determinar parâmetros de consenso que garantem o máximo rendimento com tolerância a falhas e tempo de confirmação da transação. Como vamos escalar uma rede com tais limites? Nossa abordagem é compor o blockchain de nossa rede como uma hierarquia de sub rede (Figura 1), cada uma realizando seu próprio consenso, a resultados dos quais são borbulhados até um nível mais alto, que processam esses blocos de ponto de verificação como transações ordinárias. Esta abordagem é muito semelhante às fibras da cadeia (também no uso de validadores) porém diferente por aplicarmos uma técnica de redução de dimensionalidade da localidade do hashing sensível, a fim de aumentar o tamanho do conjunto de transações. Como o efeito dessa arquitetura é reduzir a complexidade de consenso de um caso médio de O (n^3) ou quadrático por nó, usando o fato de que no Honeybadger BFT o algoritmo de consenso pode ser reduzido a O (1) por delegado[12], nossa complexidade se torna

$O(n) \rightarrow O(m): m << n$ (3)

Portanto, isso é sub linear com relação ao número de nós na sub rede.

Nesse caso, nos referimos a esses blocos de ponto de verificação como um bloco de hash sensível à localidade. Cada sub rede forma um hash de bloco sensível à localidade e a envia para a rede principal (Galáxias, descritas abaixo). Esta rede pai então trata esses blocos como transações ordinárias na próxima rodada de consenso.

Para aumentar o desempenho da rede, a escolha de vizinhos em uma sub rede precisa depender apenas da minimização da latência, já que cada nó monitora o próprio histórico de transações.

Em cada nível de escala, a mesma arquitetura auto similar de passagem de mensagem fofocada é empregada, permitindo que os mesmos nós participem em cada nível com apenas uma mudança nos recursos fornecidos por eles.

Todos os nós no Constellation podem estar "sonolentos", ou seja, eles podem entrar e sair da rede a qualquer momento. À medida que novos nós entram na rede, seus recursos são alocados para uma sub rede. Depois que esta atinge o limite para o número de transações e de segurança criptográfica (o número de facilitadores versus o número dos participantes), conforme descrito acima, a inserção de nós deve formar uma nova sub rede.

Assim, novas sub redes serão alocadas dinamicamente à medida que os membros saem e entram na rede. O objetivo dessa estrutura auto similar e nosso modelo de seleção de delegados descrito abaixo é ativar o dimensionamento automático dinâmico com base nos recursos de rede, permitindo, dessa forma, uma frequência de transferências consistente. É trivial mostrar que essa arquitetura exibe a lei de poder intrínseca de redes livres de escala, que são notórias por sua aplicação à tolerância a falhas em sistemas complexos, como a computação distribuída[13].

12M. Ben-Or and R. El-Yaniv. Resilient-optimal interactive consistency in constant time. Distributed Computing, 16(4):249?262, 2003 13Ravasz, E.; Barab´asi (2003). "Hierarchical organization in complex networks". Phys. Rev. E. 67: 026112.

Modelo delegado de seleção: Prova de Meme

A tolerância de erro em cadeias de confiança pode ser melhorada através de um sistema de reputação para selecionar delegações 14. Nós, desta forma, apresentamos a *Prova-de-Meme* (Proof-of-Meme); um método de consenso distribuído que incorpora participação histórica de um nó numa delegação selecionada. Por definição, um *meme* é uma unidade que representa uma ideia imitável ou comportamento que pode se espalhar. Assim, em nosso caso, ele representa um comportamento benevolente através da Constellation. Comportamento este que é recompensado e deve ser imitado para melhorar a reputação geral de um nó no sistema. A Prova-de-Meme é uma meritocracia, sendo a Prova de Aposta (Proof of stake) uma plutocracia.

Um meme, em nosso sentido, é um vetor característico correspondendo a cada conta de um nó; no caso mais simples, é uma matriz de valores Reais (float) usados como entrada para uma máquina de aprendizado de algoritmo

determinística. Nós seguimos os passos de REGRET, que usa características ontológicas para descrever a reputação de um nós em uma rede ; tecnicamente, esse é um produto tensor de espaços de atributo. 15. Um modelo de uma máquina determinística de aprendizado semelhante, que usa essa ontologia (*meme*) como espaço de atributo, é usado para determinar a pontuação de reputação e transitivamente a probabilidade daquele nó ser escolhido para participar do consenso.

Um *meme* é a base da pontuação de reputação, e essa pontuação dá a probabilidade de ser escolhido para consenso.

A Seleção de Delegação Probabilística tem sido estudada extensivamente no contexto de melhorar o *BFT* em mecanismos assíncronos de consenso 16. Especificamente no trabalho deles no *GURU, A. Biryukov et al.* tem mostrado que uma tolerância típica de erro de 1/3 de nós maliciosos em consensos bizantinos pode ser melhorada para até (e em alguns casos mais) 1/2. Nós incorporamos sua estrutura de validação para nossa seleção de delegação na Prova-de-Meme.

Existe de um grupo de trabalhos relacionados à Corrente-de-Confiança-Extendida (Extended Trust Chain) que soluciona o problema da resistência sybil de modelamento de reputação. Especificamente no *P. Otte*17 o algoritmo Fluxo-Rede (NetFlow) para previnir ataques sybis e uma modificação do PageRank, nomeadamente Temporal PageRank, para atualizar o estado de reputação. Constellation primeiro irá replicar o Temporal PageRank e então melhorar onde possível durante a análise de performance.

Nosso uso de pontuação baseada em reputação para seleção de delegação com a Prova-de-Meme, em conjunto com o papel de cada nó como contas individuais, permite à nossa rede, sem permissões, os benefícios de um sistema permissionado, reforçando transparência, o que incentiva bom comportamento. Todo histórico de transações e consenso, para cada nó (e transitivamente microsserviço) é publicamente notarizado. Isso permite um tipo de confiança que é negligenciado em tecnologicas existentes que reforçam bom comportamento, com taxas de transação e mecanismos de consenso desiguais.

Prova-de-Meme Explicada

Nós discutimos três campos de trabalho que solucionam problemas de conluio em consenso distribuído. Esses são todos componentes para nossa delegação de seleção, como segue. Nós seguiremos a abordagem utilizada em REGRET para desenvolver uma ontologia para um meme, que se tornará o espaço característico para nossa pontuação de reputação. Nós iremos adaptar o Temporal PageRank para atualizar o estado de reputação. Nós incorporamos a estrutura de validação de GURU para nosso método de seleção de delegação. Detalhes adicionais serão adicionados durante o processo de desenvolvimento. Construir uma ontologia de reputação, treinar um modelo determinístico para pontuação de reputação e implementar as amostras de algoritmos para seleção de delegação são componentes de nosso desenvolvimento.

Nós agora detalharemos como essa pontuação é usada para seleção de delegação.

Cada característica é representativa da utilidade de um nó para a rede (se fora notavelmente falha durante o consenso, quanta memória pode dedicar para o consenso, e assim vai...). Um meme pode ser transformado num valor numérico, passando-o para uma função, um modelo a ser treinado como descrito acima, que é como nós quantificamos a reputação de um meme. Afim de prover um mecanismo que promove novos nós a melhorarem seus memes, a probabilidade de distribuição de pontuações de memes é desenhada sobre buckets de tamanho fixo; a distribuição contorna a probabilidade de um meme com pontuação em um balde em particular seja escolhido. Da perspectiva computacional, isso provavelmente será implementado como um algoritmo de agrupamento que re-agrupa nós como uma sub-rotina. Isso deve maximizar taxas de transferências por aliviar o número de facilitadores ainda mantendo o mesmo nível de tolerância a erro e tempos de cornificação. Memes bem conceituados

serão dados preferência, mas novos memes também terão uma oportunidade de participar e construir suas reputações na rede. Registros de performance serão notarizados em corrente, então memes que provêm recursos falhos ou performem de forma adversarial terão suas reputações reduzidas, enquanto memes de alta performance e confiáveis terão suas aumentadas. Nossa ideia para seleção de delegação no caso indutivo é como segue:

1) *Consenso* é realizado

2) O bloco de *hash* é alimentado com um algoritmo determinístico que produz pontuações de memes atualizadas para cada delegado. Isso é realizado no topo da camada (*Galáxia*) que é responsável por escolher facilitadores.

3) Essa constante é usada para embaralhar nossa distribuição (mover memes entre buckets baseado na informação de suas performances).

4) O bloco de hash anterior (resultado do último consenso) é usado para ordenar os conteúdos de cada baucket o top N de cada(de acordo com uma distribuição de probabilidade) são selecionados nesse round. O meme de um participante do consenso que atuar falho ou provavelmente maliciosamente (verificável pelos registros) será ancorado no próxima eleição eleição do participante do consenso.

À medida que a reputação de um meme começa a carregar mais valor, ela substitui a necessidade de uma taxa de transação. Microsserviços podem parecer memes de boa reputação para hospedagem (hosting) de serviços, o que seria mais vantajoso do que arrecadar taxas de transações por realização de consenso. Invés de aumentar os preços quando a latência está alta, os memes com pontuação de reputação baixa serão reprimidos; suas transações processadas com baixa prioridade. Um meme nesse caso,

teria recursos (participar num consenso) assim aumentando sua taxa de transferência e resolvendo problemas de largura de banda para a rede. 13

Possua seu Meme

Constellation não terá taxas de transação de rede, em vez disso, reputação será o mecanismo para prevenir latência e ataques adversário. Afim de semear (seed) nossa rede, um incentivo de rede será providenciado na forma de recém-formados tokens, depois do bloco inicial genesis (que alocará tokens para participantes ICO) com retornos diminuídos até uma data fixa, quando a formação de tokens será cessada. Tokens recém formados serão alocados baseados em cada uma das reputações de seus memes.

Considere o cenário de um ponto no sistema de vendas, especificamente uma máquina de vendas. Uma máquina de vendas que opera em Constellation poderia ser implementada como um microsserviço que escolhe participar em consenso. A reputação de seu meme irá crescer com o tempo, e transações de tempo curto serão garantidas mesmo em situações de tráfego alto. Isso é vital, visto que uma aplicação de mundo-real necessita de uma taxa de transferência alta para prover

um serviço competitivo. Isso significa que mesmo que um dono de uma máquina de vendas tenha uma reputação de meme baixa, a reputação alta da máquina de vendas garantirá que aquele dono receba seu item expedientemente, mesmo em um período de alto tráfego.

Arquitetura Ponta à Ponta

Nossas seções anteriores tangem a arquitetura de nossa camada ponta a ponta, que agora será mais detalhada.

Estrelas

A estrela é o objeto base na Constellation. Para interagir diretamente com a rede, usuários instanciam um nó-estrela em um dispositivo e transações são emitidas por essa instância da estrela. Cada estrela contém uma corrente local que é composta de seu histórico na rede. Essa corrente local é usada para reforçar a orientação e é identificada por sua chave pública, cuja contraparte privada é usada para assinar transações. Uma estrela em si é um cliente leve para um usuário interagir com a rede e é nativamente compatível com dispositivos móveis. De qualquer forma, se uma estrela deseja, ela pode escolher participar em um consenso, o qual a permitirá que aprecie a reputação de seu meme. Se uma estrela decide participar em um consenso, ela então se junta a uma coleção de estrelas, as quais nos referimos como um Aglomerado de Estrelas (Star Cluster)

Aglomerado de Estrelas (Star Clusters)

Um aglomerado de estrelas é uma coleção de estrelas que escolheram participar de um consenso. O número total de estrelas é limitado pela fronteira superior, que será determinada após experimentos e análises estatísticas suficientes, como destacado acima. Quando atingimos esse marco, um novo aglomerado de estrelas é criado e cada um deles forma blocos de hash sensíveis à aglomeração. Esses blocos de hash sensíveis à localização são então tratados como transações ordinárias e são *picados* (hashed) por galáxias em, nomeadamente, *Buracos Negros* (Black Holes).

Galáxias

Galáxias são isomórficas no papel de validadores na Corrente-de-Confiança-Extendida (ExtendedTrustChain), ainda também, eles servem como um grupo auto escalável, provisionando recursos para novos aglomerados de estrelas (Star Clusters) e mantendo a reputação de memes. A reputação de um meme é calculada a partir da visualização de registros de performance de consenso. Metadata sobre os conteúdos de um bloco proposto, latência e falhas são enviadas com blocks sensíveis a localidades para Galáxias.

Assim que uma Galáxia recebe essa informação, a reputação do meme é atualizada, antes de uma nova amostra ser escolhida para o próximo round de consenso. Galáxias, assim como seus validadores, provem uma fonte de verdade para remover transações inválidas e decidir delegações em aglomerados de estrelas. Metadata sobre a performance da galáxia é guardada em buracos negros. À medida que as estrelas atingem um marco de reputação, elas podem ganhar o direito de funcionar como nós de galáxias.

Buracos Negros

Buracos negros são blocos sensíveis à localidade. É equivalente referir-se a eles como blocos numa corrente de blocos. Galáxias guardam o histórico inteiro da blockchain.

Contratos Inteligentes como micro serviços

Sistemas altamente disponíveis, elásticos e distribuídos prosperam numa arquitetura sem servidor. No caso de um sistema operacional distribuído, isso pode ser alcançado por uma rede de microsserviços distribuídos. Assim, na Constellation, contratos inteligentes por si são microsserviços operando em um JVM. Eles podem enviar transações, assinar contrapartes e realizar consenso. O objetivo é que microsserviços por si operem como uma estrela com um meme correspondente, provendo serviços por quantias concordadas. Eles podem servir o mesmo papel

que contratos inteligentes no Ethereum ou contrapartida, mas adicionalmente, provem lógicas mais complexas utilizando o código-base existente no ecossistema JVM. Além disso, eles podem conversar com programas externos através de uma interface RPC. Se esses micro serviços são construídos com um nível concreto de serviços de acordos, ou melhor ainda, tipos de assinatura, sua lógica composta pode ser interligada (chained) e composta em aplicações distribuídas intuitivamente. É aí que entram os MapReduce Operators. Um contrato inteligente de micro serviço pode ser desenhado para enviar e receber modelos de informação que melhoram a complexidade computacional (dada a arquitetura assíncrona) e podem ser reaproveitados para novas aplicações. Considerando a metáfora celestial acima, é válido notar que aplicações distribuídas na Constellation são constelações por si. Como cada micro serviço é uma estrela, coleções de micro serviços interligados e/ou compostos podem ser conectados por uma linha que representa uma aplicação. É trivial notar que quando desenhada, isso produz uma constelação.

Constellation como um Sistema Operacional de Blockchain

As arquiteturas-sem-servidor acima são exemplos de um sistema operacional distribuido. O objetivo de um sistema operacional é prover uma interface para utilização de recursos subjacentes do hardware. Isso é integral para desenvolvimento de aplicações, permitindo um alto nível de linguagens de programação e interfaces intuitivas ao usuário. Um sistema operacional distribuído difere que mira em prover uma interface para os recursos subjacentes de um aglomerado computacional. Como blockchains são inerentemente um sistema distribuído, escrever aplicações compatíveis com blockchain requere um sistema operacional distribuído, afim de utilizar todo o potencial de um aglomerado subjacente.

Tipicamente, sistemas operacionais requerem um programa que mantenha o estado subjacente do aglomerado. Isso é conhecido como um núcleo (kernel). É

possível construir um sistema operacional distribuído com uma arquitetura sem servidor. No nosso senário como em *MicroOs*, o sistema operacional é uma coleção de microsserviços. Extrapolando isso, de uma perspectiva de um desenvolvedor de aplicações, microsserviços são blocos de construção, assim como Legos, para aplicações compreensíveis construídas com Constellation. O objetivo disso é permitir até indivíduos não técnicos que construam aplicações usando microsserviços existentes como blocos de construção. Isso acoplado com uma interface intuitiva ao usuário provem uma rampa-integrada, sem atrito, para usuários não técnicos desenvolverem aplicações distribuídas com Constellation.

Assim como descrevemos acima, a habilidade de um individuo para perfeitamente (não de forma técnica) materializar uma ideia em um aplicativo, um poderia dizer que isso é a transmutação de energia potencial da Noosphere no mundo material. Por causa disso, nós

Decidimos nomear nossa moeda de *noOs,* na luz da energia potencial noosférica que faz nossos objetivos e ideias.

Conclusão

Nós apresentamos uma reformulação de um consenso criptografado seguro numa arquitetura moderna e sem servidor que permitirá aplicações mainstream a usarem a tecnologia blockchain. Nossa abordagem aplicada na redução de dimensões das técnicas de hashing sensíveis a localidades como um mecanismo de "dividir e conquistar", permitindo uma *Trust Chain* na escala da **Internet**. Nossa economia baseada em memes provem uma camada mais profunda de segurança criptográfica e o potencial de tornar taxas de transações obsoletas. Adicionalmente, nós provemos uma metáfora celestial para nossa arquitetura distribuída, desenhando uma correlação entre aplicações distribuídas e um auto-familiar padrão de escala observável na natureza.