**Report**:

# Soluzioni tecnologiche per la scalabilità dell'architettura blockchain.

# Analisi dettagliata di una di queste: lo sharding.

Settembre 2021

**Con la collaborazione di**:
Università di Trento

UNIVERSITÀ DI TRENTO
Dipartimento di Matematica
Laboratorio di Matematica Industriale e Crittografia

**Fornitore**:
Athilab

athilab gruppo Yoda

Attività svolta nell'ambito dell'Avviso promosso dal Ministero dell'Università e della Ricerca per la presentazione di Idee progettuali per Smart Cities and Communities and Social Innovation di cui al D.D. n. 391/Ric. del 5 luglio 2012 e ss.mm.ii. - SIN_00968 THE LEARNING METERS NETWORK: workpackage formativo del SCN_00398 - CUP J49G14000140008.

# Prefazione

Il presente documento è parte dell'attività svolta nell'ambito dell'Avviso promosso dal Ministero dell'Università e della Ricerca per la presentazione di Idee progettuali per Smart Cities and Communities and Social Innovation di cui al D.D. n. 391/Ric. del 5 luglio 2012 e ss.mm.ii. - SIN_00968 THE LEARNING METERS NETWORK: work-package formativo del SCN_00398 - CUP J49G14000140008.

Il presente documento è stato commissionato all'azienda Athilab ed è stato redatto con la collaborazione del dipartimento di Matematica dell'Università di Trento.

L'esecuzione dell'attività di Ricerca è stata svolta da personale del Dipartimento di Matematica e ha coinvolto, oltre al responsabile scientifico, studenti di dottorato e assegnisti di ricerca, tra i quali Andrea Flamini e la dott.ssa Carla Mascia.

La responsabilità scientifica dell'esecuzione dell'attività di ricerca svolta da parte dell'Università di Trento è stata affidata al Prof. Massimiliano Sala, per quanto riguarda l'azienda Athilab è stata curata dalla dott.ssa Chiara L. Ballari.

| **Versione** | **Data** | **Autore** | **Note** |
|---|---|---|---|
| 1.0 | Settembre 2021 | Athilab con il contributo dell'Università di Trento | Prima Stesura |

# Indice

# 1 Introduzione

## 1.1 Obiettivo del Documento

Fin dal 2008, anno in cui viene pubblicato il Whitepaper di Bitcoin [10], la comunità scientifica ha largamente studiato la tecnologia blockchain analizzandone le potenzialità, i limiti e i possibili casi d'uso. Uno dei maggiori obiettivi della ricerca è stato quello di studiare soluzioni finalizzate a renderla utilizzabile nei più svariati ambiti applicativi, a partire dalla creazione di nuove crittovalute, ognuna con la propria peculiarità, fino ad arrivare a blockchain per il tracciamento della filiera in svariati settori.

Dopo aver ricoperto inizialmente un ruolo di nicchia, l'utilizzo della tecnologia blockchain si sta diffondendo sempre più. Per poter gestire l'aumento di utenti e di transazioni, garantendo un consumo consono di risorse come energia, potere computazionale e memoria allocata, queste piattaforme devono essere scalabili.

**Definizione 1.** La *scalabilità* in informatica descrive l'abilità di un processo, software, organizzazione o network di crescere ed essere in grado di gestire un numero crescente di richieste. In particolare, un software, sistema o organizzazione è detto scalabile se è maggiormente adattabile ai cambiamenti di necessità dei suoi clienti o utenti.

Oggigiorno le principali piattaforme blockchain non sono sufficientemente scalabili. Per esempio, Bitcoin, con le sue 7 transazioni per secondo (txps), ed Ethereum, con le sue 15 txps, non possono competere con circuiti di pagamento come VISA o Mastercard. Questo è un problema rilevante se le crittovalute ambiscono a diventare una valida alternativa ai metodi di pagamento tradizionali. Per questa ragione, Ethereum sta pianificando di modificare il suo protocollo di consenso, il protocollo che permette ai nodi di aggiornare il registro distribuito coerentemente l'uno con gli altri. Tuttavia, la modifica del protocollo di consenso per renderlo più efficiente, non è una soluzione completa, quindi diventa necessario prendere altre contromisure per rendere una piattaforma realmente scalabile.

Nella Sezione 2 descriviamo alcuni dei concetti utili per comprendere i contenuti di questo report; nella Sezione 3 presentiamo alcune delle tecniche che vengono adottate per aumentare la scalabilità di alcune piattaforme blockchain, focalizzandoci principalmente sulla tecnica chiamata *sharding*. Infine, nella Sezione 4, forniamo una descrizione ad alto livello di alcune delle piattaforme attive o in via di sviluppo che utilizzano lo sharding per aumentare la scalabilità.

## 1.2 Definizioni

| | |
|---|---|
| Blockchain | Registro dati immutabile composto da una catena di blocchi. Ogni blocco contiene un insieme di transazioni e, escluso il primo, un puntatore (tipicamente un hash crittografico) al blocco precedente. |
| Firme digitali | Sono schemi crittografici utilizzati per verificare l'autenticità di messaggi e documenti. |
| Hash | Funzioni che mappano un input di lunghezza arbitraria in un output di lunghezza fissata. L'output prende il nome di digest o semplicemente hash. |
| Puntatore hash | Un puntatore a delle informazioni memorizzate, a cui è aggiunto il digest delle informazioni stesse. L'hash garantisce l'integrità delle informazioni puntate dal puntatore. |
| Scalabilità | L'abilità di un processo, software, organizzazione o network di crescere ed essere in grado di gestire un numero crescente di richieste. In particolare, un software, sistema o organizzazione è detto scalabile se è maggiormente adattabile ai cambiamenti di necessità dei suoi clienti o utenti. |
| Sharding | Il processo di suddivisione di una blockchain in parti più piccole chiamate *shard*. È una tecnica per aumentare la scalabilità della blockchain. |
| Smart contract | Protocolli informatici che eseguono automaticamente tutte o parti di un accordo conservato in una piattaforma basata su blockchain. |
| Soft fork | Una modifica nel protocollo che regola la blockchain che rende i blocchi di transazioni precedenti alla modifica invalidi, mentre i blocchi creati successivamente risulteranno validi anche per coloro che non aderiscono alla nuova versione del protocollo. |
| Throughput | Il numero massimo di transazioni che la blockchain può elaborare in un secondo. |

## 1.3 Acronimi

| | |
|---|---|
| CPU | Central Processing Unit |
| DAG | Directed Acyclic Graph |
| EOA | Externally Owned Accounts |
| BFT | Byzantine Fault Tolerant |
| PBFT | Practical Byzantine Fault Tolerance |
| PoS | Proof of Stake |
| PoW | Proof of Work |
| tpxs | Transaction Per Second |

# 2 Preliminari

**Definizione 2.** La *blockchain* è un registro dati immutabile composto da una catena di blocchi. Ogni blocco contiene un insieme di transazioni e, escluso il primo, un puntatore (tipicamente un hash crittografico) al blocco precedente.

L'idea di una tale catena protetta dalla crittografia risale al 1991 in un lavoro di Haber e Stornetta [5] dove è stata presentata una tecnologia utile per impedire la manomissione del time-stamp di documenti. La loro intuizione non si concretizzò fino al 2008 quando una persona, o un gruppo di persone, sotto lo pseudonimo di Satoshi Nakamoto pubblicò lo Whitepaper di un nuovo sistema di pagamento basato su comunicazioni peer-to-peer: Bitcoin.

Nakamoto utilizzò la blockchain come infrastruttura sulla quale costruire il sistema di pagamento Bitcoin, un registro pubblico dove inserire le transazioni che venivano pubblicate dai mittenti appartenenti alla rete peer-to-peer.

Il principale vantaggio dell'utilizzo della blockchain come registro dove memorizzare lo storico delle transazioni è il seguente: una volta che una transazione è inserita non può essere retroattivamente modificata senza dover scartare tutte le transazioni che sono state inserite successivamente. Infatti ogni blocco della blockchain (che può essere visto come una pagina del registro) contiene un puntatore hash al blocco precedente, e piccole modifiche alle informazioni inserite in un blocco causano un cambiamento drastico nel valore dell'hash ad esso associato. Essendo la blockchain pubblica e resistente alle manomissioni, i nodi della rete possono verificare e ispezionare autonomamente le transazioni inviate ed inserite.

Le blockchain riescono a risolvere il problema della riproduzione infinita di una risorsa digitale, infatti se un nodo $A$ è in possesso di una risorsa $r$, può scegliere di trasferirla a un altro nodo $B$ solo una volta, perdendone il possesso. Prima di autorizzare il trasferimento, i nodi della rete si assicureranno che $A$ sia in possesso di $r$, osservando lo storico delle transazioni nel registro. In particolare, $A$ è entrato in possesso di $r$ perché è riuscito a crearlo oppure perché gli è stato consegnato a sua volta da un altro nodo. Una volta che le verifiche sono state effettuate, la rete inserirà l'informazione che $B$ è il nuovo proprietario di $r$.

Gran parte degli attacchi a piattaforme blockchain utilizzate da crittovalute si basano proprio sull'utilizzo della stessa risorsa per più transazioni (*double spending*), e per questa ragione, per ostacolare questo tipo di attacco, vengono utilizzati protocolli di consenso, ossia protocolli utilizzati dai nodi per accordarsi su quali transazioni sono valide e quali non lo sono, in modo da inserire nella blockchain solo quelle valide.

## 2.1 Protocolli di consenso

Esistono diverse tipologie di protocolli di consenso, ognuno dei quali mira a fare sì che i nodi che ne prendono parte siano incentivati ad agire correttamente inserendo le transazioni in seguito a una verifica di validità. Gli incentivi possono essere di diverso tipo, spesso economico, tuttavia è cruciale che risulti più conveniente per un nodo attenersi alle

direttive del protocollo piuttosto che deviare da esso per cercare di ottenere un maggiore guadagno.

I protocolli di consenso più adottati nelle principali piattaforme blockchain attive sono di due possibili tipologie: Proof of Work (PoW) e Proof of Stake (PoS). Nonostante sia diffusa la convinzione che PoS possa potenzialmente sostituire PoW nei nuovi progetti blockchain, la comunità scientifica ha sviluppato e sta studiando altre idee diverse per raggiungere il consenso, come Proof of Authority, Byzantine Fault Tolerance e Proof of Weight. Per una panoramica più completa dei protocolli di consenso rimandiamo il lettore interessato a [11, 19].

### 2.1.1 Proof of Work

I protocolli di tipo PoW basano il loro funzionamento sull'allestimento di una competizione di calcolo tra i nodi che prendono parte al protocollo stesso, detti *miner*. In particolare ogni volta che un blocco viene pubblicato, i miner iniziano a costruire quello che sarà il loro "candidato blocco" inserendo le richieste di transazioni di cui sono venuti a conoscenza. Una volta raggiunta la massima capienza del blocco, il miner inizia a risolvere un crypto-puzzle che consiste nella ricerca di un valore, detto *nonce*, da inserire nel blocco in modo tale che l'hash del blocco creato, insieme al nonce, inizi con un numero prefissato di zeri.

Il primo miner che vince la PoW, ossia risolve il crypto-puzzle, pubblica il proprio blocco contenente il nonce che ha trovato. Gli altri nodi verificano che la soluzione che ha inserito sia corretta (l'hash del blocco abbia il numero di zeri richiesti) e che le transazioni incluse nel blocco siano tutte valide. In caso affermativo, i nodi della rete aggiornano la loro blockchain inserendo il nuovo blocco e il miner che lo ha creato incassa un premio come ricompensa per l'energia elettrica investita nella risoluzione del crypto-puzzle.

Un miner malevolo che vuole inserire transazioni non valide, per esempio una transazione che invia a due nodi diversi la stessa risorsa, deve vincere la PoW e pubblicare il proprio blocco. Gli altri miner, in seguito alla pubblicazione del blocco, eseguiranno la verifica di validità della soluzione con successo. Tuttavia, verificando le transazioni incluse nel blocco si accorgeranno che sono presenti due transazioni inconsistenti quindi, se onesti, scarteranno la soluzione restando in attesa di una nuova soluzione e di un nuovo blocco. Di conseguenza il miner malevolo fallisce nell'esecuzione di un attacco di tipo double spending e non ottiene la ricompensa che gli sarebbe spettata se avesse incluso nel suo blocco solo transazioni valide. L'attacco del miner malevolo può avere successo solo se più della metà dei miner non sono onesti e cooperano con il miner malevolo.

Per maggiori dettagli, rimandiamo il lettore interessato a [9].

### 2.1.2 Proof of Stake

I protocolli di tipo PoS si basano invece su un principio differente rispetto a quello dei protocolli PoW. I nodi della rete che sono in possesso di risorse che vengono tracciate nella blockchain sono "ricchi" solo fino a quando il sistema che tiene traccia degli averi dei nodi è affidabile. Nel momento in cui la piattaforma che tiene traccia della distribuzione delle risorse tra i nodi della rete perde credibilità anche le risorse da essa tracciate perdono valore.

Di conseguenza i protocolli di consenso PoS in generale sono strutturati nel seguente modo: i nodi che vogliono prendere parte alla creazione di un blocco congelano parte delle risorse di cui sono in possesso. Maggiore è l'ammontare delle risorse che hanno congelato per poter partecipare al protocollo di consenso, maggiore è la probabilità che vengano coinvolti, diventando così *verifier*. Se un verifier si comporta scorrettamente, non seguendo fedelmente le istruzioni del protocollo di consenso e se il comportamento malevolo viene identificato dai nodi della rete, il verifier si vedrà privato delle risorse che ha precedentemente congelato. Se invece il verifier si comporta onestamente, oppure il comportamento malevolo non viene rilevato dalla rete, le risorse congelate vengono sbloccate e rimesse a disposizione del proprietario.

Questa classe di protocolli di consenso incentiva i nodi a comportarsi correttamente anche senza l'utilizzo di ricompense da attribuire ai verifier onesti. Infatti i nodi in possesso di risorse sono incentivati a candidarsi come verifier in quanto vogliono mantenere la piattaforma attiva, altrimenti questa perde credibilità e conseguentemente le risorse in essa tracciate perdono valore. Per di più, i verifier sono incentivati a seguire le istruzioni del protocollo in quanto rischiano di perdere le risorse che hanno congelato candidandosi. Se un verifier invece persiste nel comportarsi scorrettamente si vedrà sottrarre sempre più risorse fino a quando non avrà più il diritto di candidarsi a verifier. Quindi, con il passare del tempo, la piattaforma esclude i nodi malevoli.

I protocolli di tipo PoW sono ritenuti sicuri ed affidabili (basti pensare che la blockchain che controlla Bitcoin è regolata da un protocollo di consenso PoW), tuttavia molti screditano questa classe di protocolli a favore dei protocolli PoS per almeno due ragioni: la prima è l'enorme consumo di energia elettrica da parte dei miner che cercano di risolvere il crypto-puzzle, infatti si stima che la rete Bitcoin in un anno consumi una quantità di energia maggiore del consumo annuale di uno stato come la Svezia; la seconda è la perdita di decentralizzazione della rete. Infatti, i nodi che prendono parte alla competizione per poter creare un blocco, quasi sempre scelgono di unirsi in gruppi (pool) che collaborano riuscendo così a creare più blocchi ridistribuendo le ricompense sulla base della potenza di calcolo investita da ciascun partecipante. Questo approccio permette ai miner di avere entrate più costanti nel tempo rispetto al caso in cui tutti lavorassero in modo indipendente.

## 2.2 Smart Contract

Con il termine *smart contract* facciamo riferimento a protocolli informatici che eseguono automaticamente tutte o parti di un accordo conservato in una piattaforma basata su blockchain.

Gli smart contract sono l'analogo digitale dei contratti giuridici. Due o più parti si impegnano a rispettare determinati doveri se si dovessero verificare alcune clausole. Gli smart contract sono contratti stipulati tra macchine (computer) e svolgono le operazioni richieste in automatico. In particolare, i nodi della rete che mantengono il registro valutano lo stato dei contratti salvati nelle blockchain eseguendo automaticamente le transazioni necessarie mano a mano che le clausole vengono soddisfatte.

La blockchain di Bitcoin utilizza un linguaggio di script limitato che permette la creazione di pochi ed elementari smart contract. Per esempio, su Bitcoin è possibile utilizzare schemi di firma digitale multi-signature per tutelare le parti di uno scambio commerciale. Se un utente vuole comprare un libro da un altro utente, invece che inviargli la somma corrispondente e sperare che il libro gli venga consegnato via posta, si accorda con il venditore per nominare un garante fidato da entrambe le parti che si assicuri che lo scambio avvenga correttamente. In particolare l'acquirente congela la somma dovuta al venditore in uno smart contract, e solo a quel punto il venditore invia il libro all'acquirente. Per scongelare i soldi bloccati nello smart contract è necessario che due persone (tra venditore, acquirente e garante) firmino la transazione di sblocco. Nel caso in cui sia il venditore che l'acquirente sono onesti non ci sarà bisogno di coinvolgere il garante ed entrambi firmeranno la transazione che trasferisce la somma all'acquirente. In caso di disputa che non può essere risolta dalle due parti coinvolte, il garante prenderà le parti dell'utente che si è comportato onestamente stabilendo se firmare una transazione che restituisce i fondi all'acquirente oppure che li trasferisce al venditore. In ogni caso due firme vengono raccolte ed è possibile portare a termine l'operazione commerciale in sicurezza.

Per cercare di sfruttare maggiormente le potenzialità della blockchain e degli smart contract, sono state ideate piattaforme blockchain che supportano un linguaggio di programmazione con maggiore potenzialità e più espressivo che permette di implementare smart contract più sofisticati rispetto a quelli disponibili su Bitcoin. Per esempio, Ethereum utilizza il linguaggio di programmazione chiamato *Solidity* per permettere ai suoi utenti di creare un enorme ventaglio di smart contract, rendendolo uno strumento molto versatile grazie alla varietà di applicazioni decentralizzate che supporta.

L'interesse per le piattaforme in grado di supportare smart contract complessi è cresciuto nel tempo. In particolare, la blockchain risulta essere uno strumento utile non solo per l'implementazione di sistemi di pagamento, ma anche uno strumento di supporto ad aziende o alla pubblica amministrazione. Infatti, implementando gli smart contract utili al caso d'uso in questione, la blockchain diventa un registro digitale che può essere di supporto in molti ambiti, dalla creazione sistemi di controllo dei processi aziendali o amministrativi fino al tracciamento di filiera.

## 2.3 Teoria dei grafi

In questa sezione, introduciamo la terminologia relativa alla teoria dei grafi, che sarà necessaria nella trattazione della Sezione 3.3 e della Sezione 4.3.

Un *grafo* $G$ è una coppia $(V, E)$, dove $V$ è un insieme i cui elementi sono detti *vertici*, ed $E$ è un insieme di coppie di vertici, i cui elementi sono detti *archi*.

Se gli archi sono considerati come coppie ordinate di vertici, allora $G$ è detto grafo *diretto*. Spesso i grafi diretti vengono detti anche orientati.

Un *cammino* di $G = (V, E)$ è una sequenza di vertici $v_0, \ldots, v_n \in V$ tale che, per ogni $1 \leq i \leq n$, $\{v_{i-1}, v_i\} \in E$.

Un *ciclo* di $G = (V, E)$ è un cammino $v_0, \ldots, v_n$ tale che $n \geq 3$, $v_0 = v_n$, e tutti i vertici $v_0, \ldots, v_{n-1}$ sono distinti.

Un grafo che non contiene cicli è detto *aciclico*. Un grafo diretto che non contiene cicli è detto *grafo diretto aciclico* e si indica brevemente con DAG.

Dato un grafo $G = (V, E)$, un sottoinsieme $C$ di vertici è detto *clique* di $G$ se per ogni coppia di vertici distinti $u, v \in V$, $\{u, v\} \in E$. Una clique è detta *massimale* se $C \cup \{u\}$ non è una clique, per ogni $u \in V \setminus C$.

# 3 Tecniche per aumentare la scalabilità

Esistono differenti approcci per aumentare la scalabilità delle blockchain [20]. Questi si basano su utilizzi non convenzionali delle risorse di cui la blockchain è fornita, ossia lo spazio disponibile nei blocchi per l'inserimento delle transazioni, la struttura della catena di blocchi e la rete di nodi che lavora per il mantenimento della blockchain.

In questa sezione presenteremo alcune delle tecniche che sono state adottate per risolvere il problema della bassa scalabilità.

## 3.1 Segregated Witness (SegWit)

*Segregated Witness (SegWit)* [7] è un aggiornamento del protocollo Bitcoin sviluppato nel 2015. Il concetto è stato introdotto come una soluzione al problema di scalabilità che le tecnologie blockchain affrontano ancora oggi. L'idea centrale è di riorganizzare i dati del blocco in modo che le firme non siano più posizionate insieme ai dati delle transazioni. Poiché la dimensione del blocco rimane fissata, questa soluzione permette di archiviare più transazioni in un singolo blocco, aumentando così le prestazioni in termini di transazioni inserite.

In blockchain come Bitcoin, il 65% dello spazio dei blocchi è occupato da dati riguardanti le firme digitali, quindi l'idea di SegWit è di spostare parte di queste informazioni in una side-chain a cui la blockchain principale farà riferimento.

Infatti, i dati riguardanti le firme delle transazioni sono rilevanti solo nella prima fase di verifica di validità delle transazioni stesse, quindi il fatto che tutti i dati relativi alle firme digitali vengano salvati all'interno dei blocchi una volta che le transazioni sono state accettate risulta essere ridondante e uno spreco di spazio.

SegWit è stato applicato a Bitcoin nel 2017, ma è una modifica strutturale applicabile a molteplici piattaforme blockchain che usano un'architettura simile a quella di Bitcoin.

Non tutti i dati delle firme digitali vengono rimossi, però è possibile ridurre fino al 25% lo spazio nel blocco ad esse dedicato. Creando un blocco che può contenere un maggior numero di transazioni, SegWit ha anche consentito di aumentare la velocità di transazione, dato che ci può essere un numero più grande di transazioni in movimento attraverso la blockchain. Anche se un blocco potrebbe richiedere lo stesso tempo per il processo di mining, un maggior numero di transazioni stanno venendo elaborate in esso. L'aumento della velocità di transazione ha anche contribuito a ridurre i costi delle commissioni delle transazione.

SegWit è un aggiornamento *soft-fork*, il che significa che è retro-compatibile. In altre parole, i nodi Bitcoin che non sono aggiornati per includere SegWit sono ancora in grado di elaborare transazioni. Attualmente (primo semestre 2021), la percentuale di indirizzi Bitcoin che utilizzano SegWit è intorno al 53%.

Pur essendo una soluzione potente e innovativa, SegWit ha qualche limitazione. Infatti la sua implementazione è complessa e la maggioranza dei nodi della rete deve aderire alla modifica di protocollo. Inoltre, aumentare le transazioni per blocco comporta un aumento di costo nelle comunicazioni peer-to-peer e in generale aumenta il consumo delle risorse da investire per mantenere il sistema. Infine, la side-chain in cui sono salvate

i dati che sono state rimossi dai blocchi deve essere mantenuta, tuttavia non esiste un incentivo per i nodi che lo fanno.

## 3.2 Aumento della dimensione dei blocchi

Un altro modo per risolvere i problemi di scalabilità delle blockchain attuali è quello di aumentare la dimensione dei blocchi facendo sì che possano contenere un numero maggiore di transazioni. L'aumento del numero di transazioni per blocco si ripercuote sugli utenti della piattaforma facendo diminuire il prezzo delle commissioni rendendo il servizio più conveniente. Tuttavia, come abbiamo visto nel caso di SegWit, l'aumento del numero di transazioni per blocco non è una soluzione completa se non viene combinata ad altre tecniche. L'aumento del potere computazionale necessario per fare mining causa una riduzione della decentralizzazione del sistema e l'aumento del peso di ogni blocco fa sì che lo spazio di memoria necessario a memorizzare l'intera blockchain aumenti proporzionalmente. In particolare, raddoppiando la dimensione del blocco, nel caso di Bitcoin si passerebbe da 7 txps a 14 txps, mentre nel caso di Ethereum si migliorerebbe la scalabilità passando da 15 txps a 25 txps.

Al fine di rendere il servizio davvero efficiente, bisognerebbe aumentare la dimensione dei blocchi di ordini di grandezza e questo si rifletterebbe sullo spazio di memoria di cui un singolo necessita per salvare localmente la blockchain.

Infine, come per SegWit, questa tecnica richiede un aumento della larghezza di banda per gestire le comunicazioni tra i nodi e per mantenerli sincronizzati. Mantenere i nodi il più possibile sincronizzati è essenziale nel contesto del mantenimento di un registro distribuito. Questo diventa più complesso in caso di aumento della dimensione dei blocchi, e i nodi che non hanno una banda sufficientemente larga potrebbero perdere le ricompense se altri nodi, con una migliore connessione, riescono a diffondere il loro blocco per primi. Questo scenario porta nuovamente alla centralizzazione del network in quanto i nodi con una migliore connessione risulterebbero avvantaggiati.

## 3.3 Grafi Aciclici Diretti

L'architettura classica delle blockchain può essere estesa con una più sofisticata basata su grafi aciclici diretti (DAG).

Un DAG sostituisce la struttura lineare della blockchain nella gestione dei blocchi di transazioni, permettendo la coesistenza di più filoni di blocchi. In un sistema basato su DAG, ogni vertice del grafo rappresenta una transazione. Invece di raccogliere le transazioni in blocchi, ogni transazione viene impilata sopra un'altra. Quando un nodo trasmette una transazione, c'è una piccola operazione di PoW, per evitare che vengano propagate transazioni non valide e per convalidare le transazioni precedenti.

Per aggiungere una nuova transazione, questa deve essere costruita sopra transazioni più vecchie, ossia deve fare riferimento alle precedenti. Un po' come un blocco in Bitcoin fa riferimento al suo precedente, ma in questo caso ci sono multiple transazioni referenziate.

In alcuni sistemi, un algoritmo seleziona le transazioni su cui una nuova transazione deve costruire. Queste sono transazioni ancora non confermate e prendono il nome di *punte*. Le punte con maggiore probabilità di essere selezionate sono quelle che hanno accumulato più peso, una misura del numero di conferme nel percorso verso la punta. Una volta che una nuova transazione costruisce sopra una punta, quest'ultima viene confermata.

Anche i DAG hanno un meccanismo per prevenire il double spending. È simile a quello del Bitcoin, ma senza i miner. Quando un nodo conferma le transazioni più vecchie, verifica l'intero percorso fino alla prima transazione del DAG per assicurarsi che il mittente abbia un saldo sufficiente. Potrebbero esserci percorsi diversi, ma solo uno deve essere verificato. Se gli utenti costruiscono su un percorso non valido, corrono il rischio di vedere la loro transazione ignorata. La loro potrebbe essere legittima, ma se la precedente non lo era, nessuno vorrà estendere quel particolare percorso.

Questa tecnologia porta alla parallelizzazione della validazione delle transazioni e aumenta la portata di transazioni elaborate. Non essendoci miner, gli utenti non devono necessariamente pagare commissioni per trasmettere le proprie transazioni. Inoltre, non utilizzando protocolli di consenso di tipo PoW come nel Bitcoin, il dispendio energetico è nettamente inferiore con vantaggi dal punto di visto ambientale. I DAG possono elaborare molte più transazioni al secondo rispetto alle reti blockchain tradizionali. Questo li vede particolarmente adatti nei casi d'uso dell'Internet of Things (IoT), in cui diversi tipi di dispositivi interagiscono tra loro. Tuttavia, è nota l'enorme difficoltà implementativa di una piattaforma che adotta DAG.

Esistono diversi progetti che adottano DAG, ciascuno con le proprie particolarità, per esempio SPECTRE [15], Nano [6], IOTA [12] e Blockclique [4]. Quest'ultima verrà approfondita nella Sezione 3.3. Ognuno dei progetti sopra citati ha i suoi punti di forza e debolezze a seconda di come viene costruito il grafo e a seconda di come viene implementato il protocollo di consenso. Per una visione d'insieme sull'argomento, rimandiamo il lettore a [18].

## 3.4 Off-chain State Channel

Uno *state channel* è un canale di comunicazione bidirezionale che permette a due entità di interagire tra loro. Spesso gli state channel permettono di effettuare comunicazioni che normalmente vengono tracciate sulla blockchain, al di fuori della blockchain. Questo porta con sé due principali vantaggi:

- riduzione dei costi: una minore quantità di dati da inserire comporta una riduzione del costo delle commissioni;
- tempi di iterazione: se le comunicazioni avvengono off-chain, le due parti non devono più aspettare il tempo necessario per includere una transazione in un blocco.

Esistono tre principali step nell'esecuzione di uno state channel come viene mostrato in Figura 1.

1. *Apertura dello state channel*: ogni partecipante alla comunicazione deve firmare una transazione iniziale che trasferisce soldi a uno smart contract. Questi soldi vengono congelati in modo tale che non sia possibile montare un attacco basato sul double spending sui due livelli di comunicazione: quello on-chain e quello off-chain.

2. *Interazione off-chain*: i partecipanti alla comunicazione interagiscono producendo time-stamp e firmando digitalmente le loro transazioni. Questi aggiornano lo stato della comunicazione gratuitamente e rapidamente, in quanto lo stato della blockchain non deve essere modificato.

3. *Chiusura dello state channel*: i nodi caricano sulla blockchain lo stato finale delle comunicazioni avvenute off-chain chiudendo il canale e sbloccando i fondi che erano stati congelati nello smart contract all'apertura del canale stesso.

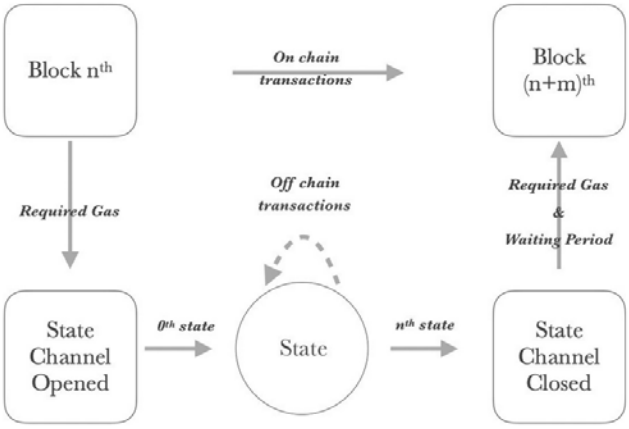


Figura 1: Flusso di uno state channel

L'apertura e la chiusura di uno state channel deve essere progettata per permettere a un nuovo stato di andare a sostituire uno precedente. È quindi necessario sviluppare dei meccanismi che assicurino che nessuna delle parti coinvolte si avvantaggi a discapito delle altre. Fino a quando i meccanismi che tutelano le parti in gioco sono sicuri e affidabili dal punto di vista teorico non ci sarà alcun bisogno di utilizzarli in quanto a tutti conviene comportasi seguendo le regole chiudendo il canale di comune accordo.

Il principale inconveniente delle implementazioni degli state channel è il fatto che i partecipanti devono poter chiudere il canale in un qualsiasi momento, ciò significa che devono rimanere online per tutto il tempo da quando lo state channel viene aperto.

Uno state channel è utilizzabile solo quando le parti in gioco non cambiano da quando il canale viene aperto e inoltre risulta conveniente utilizzarlo solo quando il numero di messaggi scambiati off-chain è alto e per un periodo lungo di tempo, infatti la sua apertura ha un costo che deve essere inferiore all'ammontare di commissioni che sarebbero state versate per caricare on-chain tutte le transazioni.

### 3.5 Concorrenza

In informatica, la *concorrenza* è l'abilità che permette a più parti di un programma, un problema o algoritmo di essere eseguite senza un ordine ben preciso, o parzialmente preciso senza andare a compromettere il risultato finale o l'output. La concorrenza quindi permette l'esecuzione in parallelo di più unità indipendentemente dall'ordine di esecuzione andando ad aumentare la velocità di esecuzione.

Nell'ambito delle piattaforme blockchain la concorrenza può per esempio essere sfruttata nell'esecuzione degli smart contract.

Nella blockchain del Bitcoin, quando un miner crea un blocco, assembla una sequenza di transazioni e calcola un nuovo stato provvisorio eseguendo gli smart contract nell'ordine in cui si verificano nel blocco. Un miner non può semplicemente eseguire questi contratti in parallelo, perché potrebbero esserci dei conflitti che porterebbero ad uno stato finale incoerente. Per le transazioni Bitcoin, è facile dire in anticipo quando due transazioni sono in conflitto, perché i dati di input e output sono dichiarati staticamente. Per gli smart contract, al contrario, è impossibile dire in anticipo se due codici sono in conflitto. I miner vengono ricompensati per ogni blocco che aggiungono con successo alla blockchain, quindi hanno un forte incentivo ad aumentare il throughput, ossia il numero di transazioni al secondo che il sistema è in grado di gestire, parallelizzando le esecuzioni degli smart contract.

Un modo per superare queste limitazioni è quello di eseguire in modo concorrenziale gli smart contract mediante un *esecuzione speculativa*. Quest'ultima è una tecnica di ottimizzazione che consiste nel fare eseguire al computer operazioni che potrebbero essere necessarie solo in un secondo tempo. Elaborando i dati prima di sapere se è davvero necessario farlo può ridurre i ritardi che si avrebbero facendo il lavoro solo dopo aver saputo se è davvero necessario o no. Se ad un certo momento del flusso di esecuzione il lavoro svolto anticipatamente si dimostra inutile, allora i risultati ottenuti verranno semplicemente ignorati.

Per maggiori dettagli si veda, per esempio, [14, 3, 13].

### 3.6 Sharding

La soluzione più promettente per ottenere scalabilità nelle piattaforme blockchain si chiama *sharding* [17, 8]. Questa tecnica consiste nel dividere la blockchain in più parti, tutte gestite da insiemi di nodi del network disgiunti, chiamati *shard*.

Questo approccio accelera il processo di raggiungimento del consenso e aumenta la portata di transazioni inserite in quanto gli shard possono validare le transazioni simultaneamente permettendo alla piattaforma di gestire un numero elevato di richieste da parte degli utenti.

Il termine sharding proviene dal database management, e identifica un tipo particolare di partizione dei database dividendoli in database di dimensioni minori, denominati shard. Gli shard sono più facilmente gestibili in termini di dimensioni dei server che li devono ospitare, mantenimento del database e permettono di accelerare il tempo di risposta alle

query diversificando le tipologie di informazioni in essi contenute. Ogni shard infatti può essere salvato in un server distinto, con la propria CPU, memoria e disco rigido.

Recentemente lo sharding ha raggiunto una grande popolarità a causa delle esigenza di gestire volume di dati sempre crescente. In particolare, la scalabilità dei database che implementano lo sharding aumenta linearmente con il numero di server che vengono messi a disposizione per il salvataggio dei dati migliorando le performance nella ricerca di informazioni, snellendo i sistemi di controllo degli accessi e di produzione di backup.

L'architettura blockchain classica (pensiamo sempre a quella del Bitcoin) è in grado di garantire alti livelli di sicurezza, tuttavia, come abbiamo già osservato, ha grandi problemi di scalabilità. Ogni nodo della rete, per ragioni di sicurezza, deve svolgere tutte le operazioni, quindi il sistema risulta efficiente come i singoli nodi della rete piuttosto che come la somma dell'efficienza dei componenti della rete. Inoltre, i protocolli di consenso, come di tipo PoS e BFT, risultano sicuri solo se sono eseguiti da una rete con un numero sufficientemente alto di nodi. Allo stesso tempo, maggiore è il numero di nodi nella rete, maggiore è anche il numero di attori malevoli che possono minare il raggiungimento del consenso. Di conseguenza, il numero di nodi che devono essere coinvolti a ogni round nella creazione di un blocco cresce, facendo crescere il numero di messaggi che devono essere scambiati, ciò inevitabilmente causa un rallentamento nel processo di raggiungimento del consenso.

Per queste ragioni, la rete che mantiene una blockchain può trarre beneficio da una sua suddivisione in shard che gli permetta di raggiungere il consenso più velocemente. Questo avviene ridistribuendo le transazioni che ogni shard dovrà elaborare senza andare a minare i livelli di sicurezza e di decentralizzazione che sono essenziali nell'utilizzo della blockchain come registro o database.

Definiamo quindi cosa si intende per sharding nell'ambito della sua implementazione su blockchain scalabili.

**Definizione 3.** Per *sharding* si intende il processo di suddivisione di una blockchain in parti più piccole, dette *shard*. Questo può avvenire in tre modi:

- *Network Sharding*: i nodi della rete sono divisi in shard ed eseguono il protocollo di consenso parallelamente;
- *Transaction Sharding*: le transazioni non vengono inviate a tutta la rete di nodi ma raggiungono solo lo shard che sarà incaricato di verificarle e caricarle;
- *Computational Sharding*: le stesse operazioni non vengono eseguite ogni volta da tutti i nodi, favorendo il calcolo parallelo o concorrente.

Queste tre caratterizzazioni dello sharding non si escludono vicendevolmente, infatti per avere un'architettura che sfrutta la divisione in shard spesso è necessario che vengano adottate almeno due di queste (se non tutte).

Diventa chiaro che alla base dell'implementazione dello sharding c'è una suddivisione dei nodi della rete in "consensus pools" di nodi e che il transaction sharding permette

agli shard di ridistribuire le transazioni in modo che shard diversi non debbano validare le stesse transazioni. Questo incrementa notevolmente la velocità di inserimento delle informazioni e rende superflua la memorizzazione di tutte le transazioni da parte dei nodi, senza però intaccare la robustezza e la sicurezza del sistema.

L'architettura basata sullo sharding è comunque soggetta a vulnerabilità, principalmente basate sul fatto che il numero di nodi in ogni shard è una frazione del numero dei nodi della rete, quindi per un attaccante risulta più semplice prenderne il controllo. Per mitigare questo rischio, il consenso all'interno degli shard è bene che venga raggiunto tramite protocolli di tipo PoS in quanto algoritmi di tipo PoW sarebbero più facilmente aggirabili da un gruppo di nodi malevoli che riescono a essere assegnati allo stesso shard.

Per eliminare errori nella elaborazione delle transazioni e per evitare il problema del double spending, le transazioni devono essere ridistribuite tra gli shard in modo che attori malevoli non possano recapitare transazioni inconsistenti a shard diversi. Un modo è quello di distribuire le transazioni agli shard sulla base dell'indirizzo del mittente.

Infine, il principale problema nell'implementazione dello sharding è la gestione delle comunicazioni tra shard diversi (*cross-shard*). Queste possono essere complesse da gestire e sicuramente risulta necessaria l'implementazione di un protocollo ad hoc per questo tipo di comunicazioni. Il protocollo agirà come un canale di comunicazione e cooperazione tra gli shard, tuttavia può causare complicazioni indesiderate in fase di implementazione.

Molti progetti stanno provando ad implementare lo sharding nell'architettura delle loro blockchain e nella Sezione 4 proponiamo una descrizione ad alto livello di alcune di queste piattaforme.

# 4 Piattaforme che adottano lo sharding

In questa sezione descriveremo ad alto livello alcune delle piattaforme che implementano lo sharding focalizzando l'attenzione sul modo in cui questo viene implementato.

## 4.1 Ethereum 2.0

Ethereum è una blockchain programmabile che è stata ideata per permettere agli utenti di ampliare il ventaglio di smart contract e operazioni presente in blockchain come Bitcoin. Questo è stato possibile definendo un linguaggio Turing-completo per l'implementazione di smart contract: *Solidity*.

Ethereum prevede due tipologie di account:

- Externally Owned Accounts (EOA): rappresentano identità esterne alla blockchain che vogliono interagire con essa (analogo degli account di Bitcoin);
- Contract Accounts: sono una combinazione di codice in Solidity e altri dati aggiuntivi e risiedono nei blocchi della blockchain.

La rete Ethereum ha un suo stato globale che è dato dallo stato dei suoi account. Lo stato degli EOA è dato dal loro saldo, mentre per quanto riguarda i Contract Accounts, il loro stato è dato dal loro saldo e dallo stato della loro memoria interna che viene aggiornato ogni volta che un account pubblica una richiesta di operazione da parte del contratto.

Quando la piattaforma di Ethereum viene lanciata 2015, il protocollo di consenso utilizzato è di tipo PoW. Il ricavo ottenuto dall'esecuzione del codice degli smart contract viene raccolto dal nodo che è in grado di creare il nuovo blocco. Quando un nuovo blocco viene creato, gli altri nodi verificano che le informazioni inserite nel blocco siano valide rieseguendo tutte le operazioni e le verifiche di validità prima di aggiornare lo stato che salvano localmente. A questo punto i nodi che prendono parte al protocollo di consenso iniziano una nuova competizione costruendo il loro nuovo candidato blocco. Questo approccio causa un enorme spreco di risorse come abbiamo spiegato in Sezione 2.1 e rende il sistema inefficiente in quanto un aumento di nodi che prendono parte al protocollo di consenso porta ad un aumento della capacità di calcolo che non si traduce in un aumento della portata di transazioni elaborate e inserite.

Il team di sviluppo della piattaforma Ethereum, consapevole di questi problemi di scalabilità, sta lavorando a un aggiornamento del sistema. Una delle modifiche principale è quella del passaggio a un protocollo di consenso basato su PoS. Il processo di migrazione al protocollo di consenso PoS è iniziato a dicembre 2020 ed è suddiviso in 3 fasi. Ci si aspetta che la seconda fase venga lanciata entro il secondo trimestre del 2022.

Tuttavia, la modifica al protocollo di consenso non si limita solo al passaggio da un consenso PoW ad uno PoS, infatti è composta da due aspetti fondamentali:

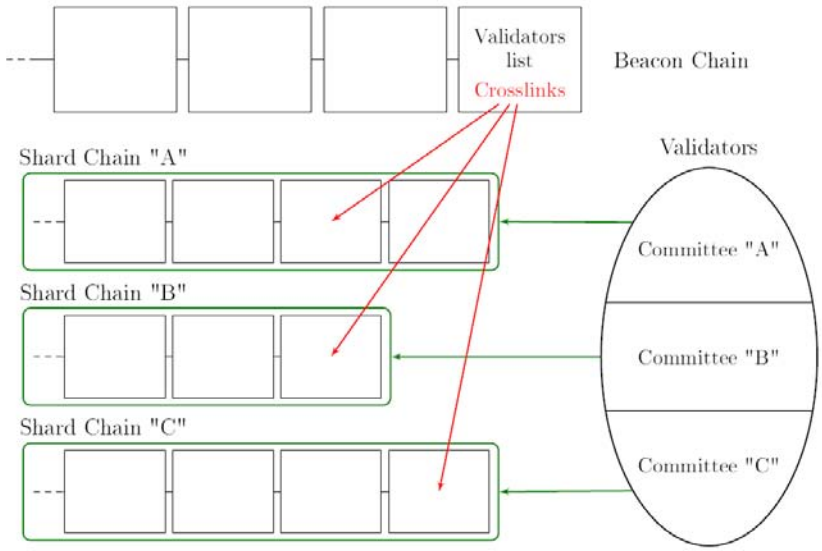


Figura 2: Architettura sharding di Ethereum 2.0

- l'adozione di Casper [1], un protocollo PoS;
- l'utilizzo dello sharding, per permettere calcolo e gestione delle transazioni in parallelo.

Queste modifiche richiedono l'introduzione di una nuova architettura che prevede l'utilizzo di una *beacon chain* e di più *shard chain*.

La beacon chain è la blockchain che ospiterà il principale meccanismo di consenso, mentre le shard chain sono catene di blocchi che raccolgono i dati degli account e le transazioni.

Per poter prendere parte al meccanismo di consenso sulla beacon chain un nodo deve congelare 32 ETH (ETH è la crittovaluta utilizzata su Ethereum). Il congelamento di tale somma renderà il nodo un *validator*. I validator appaiono in un registro gestito nella beacon chain e sono incaricati di eseguire il protocollo di consenso Casper. In particolare i validator vengono divisi pseudorandomicamente in comitati (uno per ogni shard chain) che sono incaricati di valutare le informazioni inserite nelle shard chain. Quando un numero sufficiente di validator ritiene valide le informazioni inserite in una shard chain fino ad un determinato blocco, questi creeranno un *cross-link* dal più recente blocco della beacon chain all'ultimo blocco ritenuto valido della shard chain come è possibile vedere nella Figura 2. La creazione del cross-link associato a un blocco di una shard chain fa sì che le informazioni inserite nella shard chain fino a quel blocco vengano finalizzate rendendole immutabili e disponibili all'attuazione di comunicazioni tra shard distinti.

Questa architettura fa sì che la beacon chain sia il mezzo attraverso il quale vengono gestiti i validators e viene garantita la validazione/finalità delle informazioni inserite

nelle shard chain. Le informazioni inserite nelle shard chain vengono elaborate dagli shard, gruppi di miner che svolgono le operazioni richieste dagli utenti del network, che si spartiscono le richieste in modo tale da garantire ad Ethereum una struttura ampiamente scalabile.

## 4.2 Zilliqa

Zilliqa [16] è una piattaforma blockchain progettata per raggiungere un'alta scalabilità in maniera sicura. Con una rete di 10000 nodi è infatti in grado di gestire un numero di transazioni paragonabile ai circuiti VISA o Mastercard.

Come nel caso di Ethereum, in Zilliqa esistono due tipi di account: account normali (che sono l'analogo degli EOA) e gli account contratto. Le transazioni sono sempre inviate alla rete da account normali e causano una modifica dello stato globale della blockchain.

Zilliqa prevede l'esistenza anche di due tipologie di blocco: i *Transaction block (TX-Block)* e *Directory Service block (DS-Block)*, quindi esistono due tipologie di blockchain: una Directory Service blockchain che raccoglie informazioni riguardanti i nodi che prendono parte al protocollo di consenso e metadati riguardanti i minatori, e la Transaction blockchain che raccolgono le transazioni su cui è stato raggiunto il consenso. Esistono due tipologie di TX-Block: i micro-block e i final-block. Questi svolgono due compiti diversi all'interno del protocollo di consenso però basti sapere che i final-block saranno i blocchi di cui è composta la Transaction blockchain.

Al fine di definire la struttura della rete su cui si basa Zilliqa (DS committee e shard) iniziamo a descrivere il DS committee in quanto sarà incaricato di assegnare gli altri nodi ai vari shard.

Il Directory Service committee è eletto tramite una competizione di tipo PoW tra i membri di tutta la rete. I restanti nodi parteciperanno ad un'altra competizione di tipo PoW per poter essere ridistribuiti tra gli shard. La redistribuzione dei nodi tra gli shard avviene sulla base delle soluzioni al crypto-puzzle che hanno sottomesso. I nodi appartenenti allo stesso shard sono anche classificati sulla base della qualità della soluzione che hanno trovato ed il primo classificato viene nominato leader dello shard.

Ogni shard è composto da almeno 800 nodi in modo tale che la probabilità che più di un terzo dei nodi sia malevolo sia minore di $10^{-6}$.

Zilliqa divide le richieste di transazioni tra gli shard in modo tale che una stessa transazione non venga elaborata da più shard. L'assegnazione di una transazione a uno shard è effettuato sulla base dell'indirizzo di chi invia la transazione. Questo approccio aumenta notevolmente la portata nell'elaborazione delle transazioni e semplifica la verifica da parte dei nodi che si devono assicurare che non venga effettuato double spending. Il fatto che le transazioni inviate da un account siano validate dallo stesso shard rende più semplice la ricostruzione del suo storico di transazioni in uscita. Inoltre ad ogni transazione è associato un nonce scelto dal nodo che invia la transazione. Una transazione con un nonce uguale o minore al nonce dell'ultima transazione inserita verrà scartata.

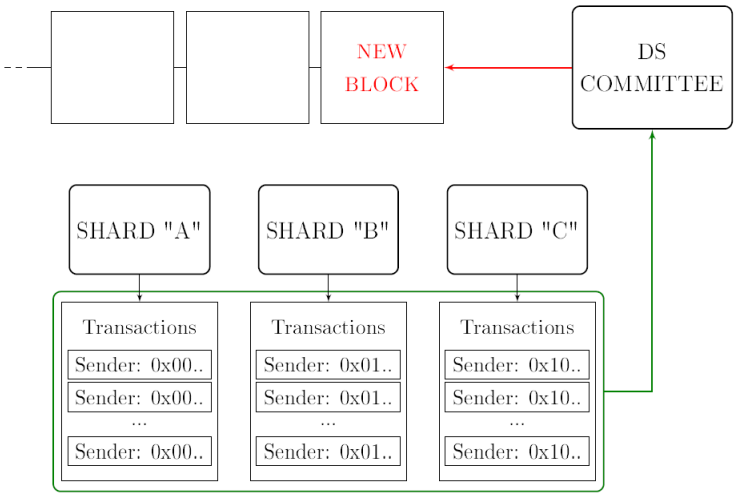


Figura 3: Architettura sharding di Zilliqa

Ogni shard raggiunge il consenso su un blocco di transazioni utilizzando il protocollo di consenso Practical Byzantine Fault Tolerance (PBFT) [2]. Se il leader è onesto le tre fasi del protocollo (Pre-prepare, Prepare e Commit) porteranno alla creazione di un certificato per il nuovo blocco, altrimenti se il leader è malevolo e mette in stallo il processo di raggiungimento del consenso, i nodi dello shard possono proporre la sostituzione del leader con il successivo nella classifica.

Una volta che lo shard ha raggiunto il consenso sul nuovo blocco, lo invierà al DS committee attuale come è possibile vedere in Figura 3. Il DS committee raccoglie i blocchi creati dagli shard e raggiunge il consenso su un final-block, comunicandolo agli shard che aggiornano lo stato degli account coinvolti.

Il linguaggio di programmazione per smart contract di Zilliqa non è Turing-completo (a differenza di Solidity) ed è progettato per fare leva sull'architettura della rete che viene diviso in shard e sul modo in cui le transazioni sono assegnate agli shard stessi. Zilliqa si assicura che data una transazione, solo uno shard dovrebbe investire potenza computazionale per inserirla. Questo approccio rientra nella categoria del computational sharding (vedi Definizione 3).

## 4.3 Blockclique

Come è stato anticipato nella Sezione 3, Blockclique utilizza grafi diretti e aciclici (DAG) per migliorare la scalabilità della blockchain. Inizialmente questa tecnica era stata introdotta per evitare lo sharding, ma successivamente il progetto Blockclique è stato sviluppato per provare che le due tecniche (DAG e sharding) possono portare a ottimi risultati se combinate.

Blockclique prevede una suddivisione dei nodi in *thread* che lavorano in parallelo e producono blocchi che sono inseriti all'interno di un DAG come nuovi vertici. Il grafo

contenente tutti i blocchi che sono stati creati nel tempo deve rispettare due regole interne:

- *thread incompatibility*: i blocchi creati da uno stesso thread devono creare una blockchain;
- *grandpa incompatibility*: i nuovi blocchi creati dai thread devono prendere in considerazione i blocchi recentemente creati dagli altri thread attraverso l'utilizzo di puntatori hash.

Le nuove transazioni vengono ridistribuite ai vari thread sulla base del mittente delle transazioni, mentre il protocollo di consenso garantisce la compatibilità delle transazioni inserite in thread diversi. Il consenso viene raggiunto sulla base della struttura del grafo diretto e aciclico ed è raggiunto automaticamente sulla clique di blocchi (block clique) massimale, ossia quella che ha richiesto la massima quantità di lavoro da parte dei nodi della rete.

Per poter identificare la clique massimale in modo efficiente, i nodi della rete salvano il bilancio di ogni indirizzo e il grafo viene aggiornato salvando soltanto i blocchi più recenti. In particolare, un blocco viene rimosso quando appartiene solo a clique di piccole dimensioni oppure quando ha un grande numero di discendenti all'interno della clique massimale.

Le simulazioni effettuate dagli accademici che hanno progettato Blockclique (non è una piattaforma attiva attualmente) suggeriscono che i parametri della sua architettura possano essere scelti in modo tale che la piattaforma raggiunga ottimi livelli di scalabilità. Tuttavia, questo modello impone numerosi vincoli sulla tipologia di smart contract che possono essere supportati dalla piattaforma, come per esempio la memorizzazione di informazioni appartenenti a blocchi cronologicamente molto datati indietro nel tempo oppure la risoluzione di incompatibilità causate da un ambiente concorrenziale.

# 5 Conclusioni

L'architettura blockchain sfrutta le caratteristiche di una rete informatica di nodi e consente di gestire e aggiornare, in modo immutabile e sicuro, un registro contenente dati e informazioni (per esempio, transazioni) in maniera aperta, condivisa e distribuita senza la necessità di un'entità centrale di controllo e verifica. Dal 2009, la blockchain ha ricevuto un forte interesse, per le sue potenzialità e per i vari ambiti applicativi: crittovalute e settore finanziario, notarizzazione di dati, tracciamento di beni, memorizzazione di flussi di dati generati da sensori.

Uno dei problemi principali da affrontare per rendere questa tecnologia fruibile su larga scala e in sempre più campi applicativi è quello della scalabilità.

La scalabilità della blockchain può essere valutata utilizzando due metriche: il throughput, ossia il numero massimo di transazioni che la blockchain può elaborare in un secondo, e la latenza, ossia il tempo necessario per confermare che una transazione sia stata inclusa nella blockchain. Una blockchain viene generalmente definita scalabile se all'aumentare del numero di partecipanti alla rete o di transazioni nel sistema non diminuisce il throughput né aumenta la latenza.

Negli ultimi anni, sono state proposte diverse soluzioni per ottenere blockchain scalabili. Alcune di queste soluzioni, come l'uso di grafi aciclici diretti (DAG), l'off-chain state channel, e lo sharding, provano a migliorare il throughput, facendo gestire le transazioni solo a una porzione della rete, invece che all'intera rete. Tuttavia, la soluzione off-chain è più soggetta a fork, mentre implementare una piattaforma che adotta DAG è estremamente complicato. Altre soluzioni prevedono invece l'aumento della dimensione del blocco o la diminuzione dei dati diversi dalle transazioni da salvare nei blocchi (SegWit).

La tecnica che sembra essere più promettente è lo sharding, in quanto risulta essere una buona soluzione per i problemi sia di prestazione sia di scalabilità. Uno schema di sharding suddivide l'elaborazione delle transazioni tra gruppi più piccoli di nodi, chiamati shard. Di conseguenza, gli shard possono lavorare in parallelo per massimizzare le prestazioni e migliorare il throughput, richiedendo al tempo stesso un sovraccarico di comunicazione, calcolo e archiviazione significativamente inferiore, consentendo allo schema di funzionare in sistemi di grandi dimensioni. La portata di transazioni che può elaborare cresce linearmente con il numero di nodi della rete che accettano di partecipare al protocollo di consenso. Poiché le piattaforme blockchain hanno bisogno di essere scalabili proprio per gestire l'aumentare del numero di utenti e quindi delle transazioni, è ragionevole credere che un approccio basato sullo sharding possa essere risolutivo se calato in un'architettura snella e parametrizzabile sulla base delle esigenze della rete.

# 6 Riferimenti bibliografici

[1] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[2] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[3] Thomas Dickerson, Paul Gazzillo, Maurice Herlihy, and Eric Koskinen. Adding concurrency to smart contracts. *Distributed Computing*, 33(3):209–225, 2020.

[4] Sébastien Forestier, Damir Vodenicarevic, and Adrien Laversanne-Finot. Blockclique: scaling blockchains through transaction sharding in a multithreaded block graph. *arXiv e-prints*, pages arXiv–1803, 2018.

[5] Stuart Haber and W Scott Stornetta. How to time-stamp a digital document. In *Conference on the Theory and Application of Cryptography*, pages 437–455. Springer, 1990.

[6] Colin LeMahieu. Nano: A feeless distributed cryptocurrency network. *Nano [Online resource]. URL: https://nano. org/en/whitepaper (date of access: 24.03. 2018)*, 16:17, 2018.

[7] Eric Lombrozo, Johnson Lau, and Pieter Wuille. Segregated witness (consensus layer). *Bitcoin Core Develop. Team, Tech. Rep. BIP*, 141, 2015.

[8] Alessio Meneghetti, Tommaso Parise, Massimiliano Sala, and Daniele Taufer. A survey on efficient parallelization of blockchain-based smart contracts. *Annals of Emerging Technologies in Computing (AETiC)*, 3(5), 2019.

[9] Alessio Meneghetti, Massimiliano Sala, and Daniele Taufer. A survey on pow-based consensus. *Annals of Emerging Technologies in Computing (AETiC), Print ISSN*, pages 2516–0281, 2020.

[10] Satoshi Nakamoto. Bitcoin whitepaper. *URL: https://bitcoin. org/bitcoin. pdf-17.07. 2019)*, 2008.

[11] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1):101–128, 2018.

[12] Serguei Popov. The tangle. *URL: http://www.descryptions.com/Iota.*

[13] Vikram Saraph and Maurice Herlihy. An empirical study of speculative concurrency in ethereum smart contracts. In *Tokenomics*, 2019.

[14] Ilya Sergey and Aquinas Hobor. A concurrent perspective on smart contracts. In *International Conference on Financial Cryptography and Data Security*, pages 478–493. Springer, 2017.

[15] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: a fast and scalable cryptocurrency protocol. *IACR Cryptol. ePrint Arch.*, 2016(1159), 2016.

[16] Zilliqa Team et al. The zilliqa technical whitepaper. *Retrieved Sept*, 16:2019, 2017.

[17] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.

[18] Qin Wang, Jiangshan Yu, Shiping Chen, and Yang Xiang. Sok: Diving into dag-based blockchain systems. *arXiv preprint arXiv:2012.06128*, 2020.

[19] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, 7:22328–22370, 2019.

[20] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to scalability of blockchain: A survey. *IEEE Access*, 8:16440–16455, 2020.