一、区块链的概念、特征和具体要素

概念

区块链是一个交易数据库，在网络中的众多计算机之间更新和共享。每次添加一组新交易时，即称为一个“区块”，区块链由此而得名。

特征

去中心化

区块链中的去中心化是指将控制和决策从集中式实体（个人、组织或团体）转移到分布式网络。 去中心化区块链网络使用透明度来降低参与者之间对信任的需求。 这些网络还会阻止参与者以降低网络功能的方式相互施加权威或控制。

不变性

不变性意味着某些东西无法变更或更改。 一旦有人将交易记录到共享总帐中，任何参与者都不能窜改交易。 如果交易记录包含错误，您必须新增新的交易以修正错误，并且这两个交易对网络都是可见的。

共识

区块链系统建立了关于参与者同意记录交易的规则。 只有当网络中的大多数参与者同意时，您才能记录新交易。

具体要素

分布式总账

分布式总帐是区块链网络中存放交易的共享数据库，例如团队中每个人都可以编辑的共享档案。 在大多数共享文本编辑器中，任何拥有编辑权限的人员都可以删除整个档案。 但是，分布式总账技术对于谁可以编辑以及如何编辑制定了严格的规则。 一旦记录了条目，就无法将其删除。

智能合约

公司使用智能合约来自行管理商业合约，而无需第三方的协助。 它们是存放在区块链系统上的程序，在满足预定条件时会自动执行。 他们会执行 if-then 检查，以便可以自信地完成交易。 例如，物流公司可以设定一个智能合约，一旦货物到达港口，就会自动付款。

公有密钥加密法

公有密钥加密法是一种安全功能，可唯一识别区块链网络中的参与者。 该机制为网络成员生成了两组密钥。 一个密钥是网络中每个人共有的公有密钥。 另一个是为每个成员独有的私有密钥。 私有密钥和公有密钥可协作解锁总帐中的资料。

二、当前存在的问题

缺失安全性

Web2的服务端架构通常利用API端点操作MySql、MongoDB、Redis等数据库，服务运行在有限的节点中，无论是容器主导的微服务还是云服务，节点数量都比较有限，虽然有Kubernetes等编排工具的调度机制保证指定数量的Service运行，但是使用DDoS使服务处于不可用状态还是相对简单。开发者需要更多的精力做安全性方面的考虑。

用户缺失对账号的所有权

Web2应用中，账号的所有权在服务创建者手中，所谓的账号只是服务创建者数据库的一系列条目，例如你有一个游戏账号，如果游戏创建者删除你的账号或者你不再玩这个游戏，你将丢失投资到游戏上的价值。

平台和内容创作者的关系严重失衡

OnlyFans 是一个由用户生产内容的成人网站，拥有 100 多万内容创作者，其中许多人将该平台作为他们的主要收入来源。 2021 年 8 月，OnlyFans 宣布了禁止色情内容的计划。 这个公告在平台创作者中引发了愤怒，他们感觉自己在帮助创建了平台后被剥夺了收入。 在遭遇强烈反对之后，这个决定很快被推翻。 尽管创作者赢得了这场战斗，但它突显了 Web 2.0 创作者的一个困境：如果离开一个平台，就会失去在平台积攒的声誉和关注。

账户身份问题

通常，你需要为使用的每个平台创建一个帐户。 例如，你可能有 Twitter 帐户、YouTube 帐户和 Reddit 帐户。 想要更改你的显示名称或个人资料图片？ 你必须在每个帐户中执行此操作。 在某些情况下，你可以使用社交媒体帐户登录，但这会带来一个常见的问题—审查。 只需点一下，这些平台就可以封锁你的整个线上生活。 更糟糕的是，许多平台要求你给他们提供你的个人识别信息才能创建帐户。

缺乏原生支付功能

Web2 的支付基础设施依赖于银行和第三方支付机构，这就把没有银行账户或碰巧生活在某些“不好”国家/地区的人排除在外。 而且如果中央数据库遭到入侵，交易双方都可能遭受损失。

三、引进区块链的意义、作用和优势

进阶安全

区块链系统提供现代数字交易所需的高水平安全防护与信任。 人们总是担心有人会操纵基础软件来为自己制造假币。 但是，区块链使用加密法、去中心化和共识这三个原则来建立一个几乎不可能被窜改的高度安全的基础软件系统。 不存在单一故障点，且单一用户无法变更交易记录。Dapps使用区块链来存储数据，这些数据会分散到网状网络中潜在的数百或数千个节点中，因此即使黑客DDoS攻击了大部分区块链网络，应用程序仍将保持安全和可用。

解决所有权问题

Web3允许通过非同质化代币 (NFT) 直接拥有所有权。 任何人甚至是游戏创作者，都没有权力剥夺您的所有权。 而且，如果您停止玩这个游戏，您可以在公开市场上出售或交易您的游戏内物品并收回它们的价值。

抗审查

在 Web3，您的数据位于区块链上。 当您决定离开一个平台时，您可以将您的声誉带走，将其带进另一个更符合您的价值观的平台。Web 2.0 需要内容创作者信任平台不会更改规则，但抗审查则是 Web3 平台的原生特性。

解决身份问题

Web3 允许你使用以太坊地址和以太坊域名服务配置文件控制你的数字身份，从而解决了这些问题。 使用以太坊地址可以提供跨平台单点登录，这种登录安全、抗审查并且匿名。

原生支付功能

使用区块链进行交易无需依赖银行和第三方交易机构，具有原生支付功能，Web3 使用诸如以太币之类的代币直接在浏览器中汇款，不需要受信任的第三方。

提升效率

企业对企业的交易可能需要大量时间，并且可能会造成营运瓶颈，尤其是在涉及合规和第三方监管机构时。 区块链中的透明度和智能合约，会使此类商业交易更快且更高效。

加速稽核

企业必须能够以可审计的方式安全地产生、交换、封存和重建电子交易。 区块链记录在时间顺序上是不可变的，这意味着所有记录总是按时间排序。 这种数据透明度将能加快审计处理速度。

四、具体实现过程

这部分以一个Web3的Todo App为例子

使用技术、框架概述

前端：React、Vite、TypeScript

React是最流行的前端框架，由Facebook开源，有着庞大而活跃的社区。Vite由Vue.js的作者EvanYou开源，最新版Vite在build时使用Rust编写的swc和Golang编写的esbuild进行文件转译。TypeScript是Microsoft开源的语言，是JavaScript的超集，JavaScript是世界上最流行和广泛使用的语言，区块链领域也大量使用JavaScript，TypeScript为JavaScript引入了类型系统，在近几年越来越流行。

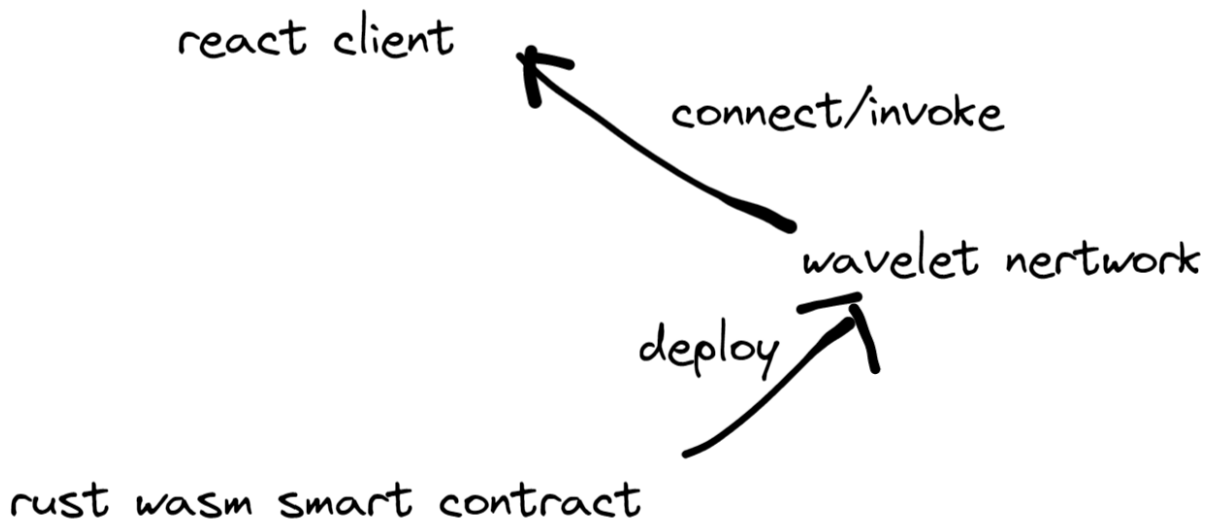
后端：Rust、WebAssembly

Rust是Mozilla开源的高性能内存安全语言，性能与C相近，此外利用所有权、Pin等特性保证内存安全、多线程安全，其强大的编译器可以保证代码质量普遍较高，不像C和CPP一样，社区里面充斥着垃圾。
WebAssembly最初作为JavaScript的补充，虽然现代JavaScript的性能已经相当高了，主流浏览器也都为JavaScript引入了JIT技术，但是在处理超多节点数量的可视化、深度学习、三维渲染等计算密集型工作时，还是有点吃力，于是提出了wasm规范，即将其它语言编写的代码编译成可供Javascript调用的可移植、体积小、加载快的二进制格式，其中最热门的选项是Rust语言。在近两年，人们发现WASM的运行在沙箱化执行环境、体积小、可移植等特点完美契合Kubernetes中Pod的特性，传统的Docker容器里面有一个简化版的Linux系统，WASM中则不然，此外WASM的沙箱环境也更加安全。Docker官方已经推出了Docker+WASM的beta版本。

部署：Wavelet

Wavelet是一个用于部署健壮、可扩展、去中心化的 WebAssembly 应用程序的区块链网络。我们使用Rust编写一个个函数，其实相当于传统Http Server的一个个API端点，在Wavelet上面非常像云函数。

架构图



具体过程

这部分确实不知道怎么写，感觉没有什么技术含量，唯一能想到的是放一些代码片段上去。

前端构建

使用tailwindcss作为样式库，react-router构建前端路由，framer-motion制作动画效果，使用Wavelet Js Client连接Wavelet服务，调用运行在Wavelet network上的智能合约函数。

连接初始化

```
import {Wavelet} from "wavelet-client";

const client = new Wavelet('https://testnet.perlin.net');
const wallet = Wavelet.loadWalletFromPrivateKey("PRIVATE_KEY");

const contract = new Contract(client, "CONTRACT_ADDRESS");
await contract.init();
```

轮询监听区块链的变化并进行同步

```
await client.pollConsensus({
    onRoundEnded: msg => {
        (async () => {
            await contract.fetchAndPopulateMemoryPages();
            // handle...
        })();
    }
});
```

后端构建和部署

Rust中使用smart_contract crate编写智能合约，打包成wasm，部署到Wavelet上

智能合约代码

```
use std::error::Error;

use smart_contract_macros::smart_contract;

use smart_contract::log;
use smart_contract::payload::Parameters;
use std::collections::VecDeque;

struct Todo {
    content: String,
    done: bool
}

struct TodoList {
    logs: VecDeque<Todo>
}

const MAX_LOG_CAPACITY: usize = 20;
const MAX_MESSAGE_SIZE: usize = 250;

fn prune_old_todos(list: &mut TodoList) {
    if list.logs.len() > MAX_LOG_CAPACITY {
        list.logs.pop_front();
    }
}
```

```
#[smart_contract]
impl TodoList {
    fn init(_params: &mut Parameters) -> Self {
        Self { logs: VecDeque::new() }
    }

    fn add_todo(&mut self, params: &mut Parameters) -> Result<(), Box<dyn Error>> {
        let todo = Todo { content: params.read(), done: false };

        if todo.content.len() == 0 {
            return Err("Message must not be empty.".into());
        }

        if todo.content.len() > MAX_MESSAGE_SIZE {
            return Err(format!("Message must not be more than {} characters.", MAX_M
        }

        self.logs.push_back(todo);

        prune_old_todos(self);

        Ok(())
    }

    fn remove_todo(&mut self, params: &mut Parameters) -> Result<(), Box<dyn Error>>
        let target:usize = params.read();
        if target < self.logs.len() {
            self.logs.remove(target);
        }
        else {
            return Err("Index out of bounds.".into());
        }
        Ok(())
    }

    fn toggle_todo(&mut self, params: &mut Parameters) -> Result<(), Box<dyn Error>>
        let target:usize = params.read();
        if target < self.logs.len() {
            let target_ref = &mut self.logs[target];
            target_ref.done = !target_ref.done;
        }
        else {
            return Err("Index out of bounds.".into());
        }
        Ok(())
    }

    fn get_todos(&mut self, _params: &mut Parameters) -> Result<(), Box<dyn Error>>
        let mut todos = Vec::new();

        for todo in &self.logs {
            todos.insert(0, format!("<{}> {}", todo.content, todo.done));
        }

        log(&todos.join("\n"));
```

```
        Ok(())
    }
}
```

总结

变革生产关系，降低社会协作成本，使区块链的研究意义凸显。在区块链的推动下，区块链在各行各业的解决方案层出不穷，我们正处于去中心化的浪潮中。