

Internet Computer^β 研究院 报告

Internet Computer 的全文介绍、
总结和报告

时间：2021.09.22





目录 CONTENT

Internet Computer^β

1. 概要
2. 技术方向
3. 代表性项目
4. 展望



01.

概要

Outline

- 01.1 Internet Computer 介绍
- 01.2 与其他区块链的区别
- 01.3 整体趋势和展望

01.1 Internet Computer 介绍

- Internet Computer 的前身

在介绍 Internet Computer 之前，不得不提一下它的前身——[Dfinity](#)，Dfinity 是由 Dominic Williams 在 2016 年成立，立志要打造一个去中心化的“互联网计算机”，来对付诸如 Google、Amazon 等巨头公司垄断的技术领域。

- Internet Computer Association (ICA)

除了上面提到的 Internet Computer 外，Dfinity 还发起了 Internet Computer Association (ICA) 组织，整个 Internet Computer 生态由 ICA 和 Dfinity [共同管理](#)。

01.1 Internet Computer 介绍

- Internet Computer 名字的由来
 - 起源于 Internet：话说 Dfinity 上有一个非常热门的视频（区块链奇点），整个视频都在介绍互联网的来历，而由于互联网是区块链技术整个大爆发的“奇点”，所以取名叫 Internet Computer，作为这个奇点之后的又一里程碑。
 - “名字”不好听：Dfinity 是区块链领域中为数不多的几个换了 Token 名字的（从 DFN 改名为 ICP）。可能 Dfinity 的老大觉得 Dfinity 的名字格局太低，念起来太low，所以改名叫 Internet Computer。

另外 Internet Computer 的缩写为 “IC”，下文为了简略，用 IC 来表示 “Internet Computer”。

01.1 Internet Computer 介绍

- 什么是 IC (Internet Computer) ?

建议不了解 IC 是什么的先看下[概念](#)和[介绍](#)。

一句话概括来说，IC 是**对抗垄断互联网行业巨头的分布式容器化应用网络**。为啥没有去中心化这个词，后面会说。

01.2 与其他区块链的区别

- 和其他区块链有啥不同？

介绍了 IC 的前身和来历后，下面来谈谈 IC 和其他区块链（特别是公链）有啥不同。IC 和其他区块链的区别，至少有以下几点不同：

- I. 访问形式的不同：目前大多数的区块链智能合约（BTC、ETH）都只能提供 API 接口访问，而 IC 不仅可以通过 API 接口访问，还可以通过 HTTP/HTTPS 协议进行访问（底层基于 IP 协议的优势之一）。
- II. 隔离的安全性：IC 的每一个 Canister（智能合约）都是一个计算单元，而且由于设计的原因，可以完全不依赖外界的任何程序或者安装包就能独立运行，其他区块链多少会用到指定平台的插件或者依赖程序，例如 EVM。

01.2 与其他区块链的区别

- III. 语言扩展性和可操作性：大部分的区块链语言都限定于一到两种作为其主要的智能合约语言，IC 底层基于 WebAssembly，理论上可以无限扩展各类语言编译成 Wasm 到 IC 上运行。
- IV. 验证的快速性：IC 利用 Chain Key 技术可以做到验证的快速性，也是 IC 官方宣传它对比其他区块链的一个主要“优势”。简单来说，Chain Key 利用 48 字节的公钥，通过这个公钥可以验证和计算整个 IC 区块链，Chain Key 还有一些其他的特性，此处不再啰嗦，感兴趣的可以参照[这里](#)。

01.2 与其他区块链的区别

- V. 社区的开放性：和一些主流的老牌区块链不同的是，IC 可以让每个 Token 持有者都有机会参与到网络的治理当中，增加了网络的多样化和开放性。对比某些区块链只有开发者社区能决定是否对网络进行升级/扩容/分叉，这个优势就非常大啦。
- VI. 自治性和网络的无限扩容性：除了让每个参与者都能参与网络的治理，IC 还让决策权“自动化”。通过 NNS（Network Nervous System）系统，对于社区的提案满足一定条件自动批准或拒绝。另外还可以对网络内的节点或子网、计算资源进行扩容，并且没有上限。对于其他大部分区块链来讲，对其网络也能进行无限“扩容”，但是由于信息缺少开放性和可参与性，没有外界约束，容易高估自己对于网络的治理能力，从而缺乏“自治性”。

01.2 与其他区块链的区别

除了以上6点不同外，IC和其他区块链还有许多方面的不同之处，由于篇幅的关系，此处就不一一列举了，感兴趣的同学可以去彭博社的[这篇采访](#)里去了解。

01.2 与其他区块链的区别

可能细心的同学会发现，前面介绍的都是 IC 对比其他区块链的“优势”，那有人可能会问，IC 的劣势在哪里？由于 IC 官方喉舌不会大肆宣传自己区块链的“缺点”，以下总结一些我个人对于 IC 区块链缺陷的看法：

- 不够去中心化

IC 从主网上线到现在一直和公链行业的老大哥以太坊作对比，但是 IC 对比其他区块链（例如以太坊）其实更像“联盟链”，或者说是弱中心化的区块链，为啥这么说捏？下面从不同角度分别举例：

01.2 与其他区块链的区别

I. 按通证持有情况来看

IC 官方[宣称](#)团队在主网上线后不会持有超过 50% 的投票权以维持网络的完全“去中心化”。首先，NNS 的投票机制表面上是一种“少数服从多数”的决策，看起来很合理，但是少数与多数之间没有建立相应的制衡与博弈机制，也就会变成谁拥有的票数多谁说了算。一旦这种情况成立，退一万步说，即使 IC 官方不作恶，一小部分长期持有大量投票权的大户，也可以私下联合提交提案，让“多数服从少数”。从这个角度看，IC 就不是完全“去中心化”滴。

01.2 与其他区块链的区别

II. 按网络活跃度来看

网络活跃度是另一个需要考虑的因素，NNS 能够做到 IC 宣传的“去中心化”的前提是，NNS 有足够多的参与者发起提案与投票，并且大部分参与者都需要有一定比例的投票力量（voting power），而且还需要相互制衡，以上这些条件都满足了，才能达到理想的去中心化效果。反之，网络只有一小部分人发起提案和参与，就和去中心化相差甚远了。

01.2 与其他区块链的区别

III. 从网络安全角度来看

NNS 的一个特点是，你可以“跟随”其他人进行投票，再加上 NNS 对外提供接口和命令行程序，从计算机和网络工程的角度来讲，你可以创建多个账户，再通过调用 NNS API 让这些账户都 follow 一个或多个匿名账户，再由这些匿名账户发起或参与某个提案，对整个网络进行攻击，从而出现一种“少数服从多数”的表象。

01.2 与其他区块链的区别

IV. 从DSS的角度来看

DSS 全称是 Decentralization, Scalability, and Security，又被称为区块链行业共识的“不可能三角”。前面提到过，IC 可以通过 NNS 无限扩容网络，也可以用 Chain Key 技术来提升整个网络的安全性，从 DSS 的角度来说，那么剩下的“去中心化”自然没办法完全满足啦。

01.2 与其他区块链的区别

- 不完全开源

除了爱拿以太坊作对比，IC 官方还喜欢抨击垄断当今互联网的巨头。讽刺的是，IC 官方承诺的开放源代码到现在还有一部分没开源，例如 DFX CLI。

01.3 整体趋势和展望

- 整体发展趋势和展望

最后，来对 IC 作个整体分析：先前我们提到过，IC 的目标是打破当今互联网的垄断局面（来源可以参考[这里](#)，各个巨头分别垄断软件/硬件/数据），为什么 IC 想要打破垄断的局面？我觉得以下原因值得考虑：

1. 让数据和权利回归个人

这个不用多说，垄断能够成立就是因为分配的失衡。

2. 自己成为新的垄断者

IC 不想让其他各巨头独占鳌头，自己也想进来分一杯羹。

3. 让互联网以区块链的形式展现

这个目的最有可能，从 Dominic Williams 的推文对区块链的彩虹屁中可以看出一些线索。

01.3 整体趋势和展望

我们假设“原因3（让互联网以区块链的形式展现）”成立。下面我们来分析一下 IC 之前做过的相关努力，以及未来需要做哪些事情来达成这个目的（相关的来源可以见 [Roadmap](#)）。

过去做过哪些事情：

- 开发新一代的区块链互联网应用，如：CanCan，LinkedUp。
- 快速融入新的互联网技术，如：用 WebAuthn 作身份验证，用 Wasm 作为 Canister 的执行语言。
- 使用 IP（Internet Protocol）作为 IC 的底层设计。

.....等等。通过上面几个例子，与互联网的结合，可以使目标更近一步。

01.3 整体趋势和展望

未来需要做哪些事情：

- 与以太坊、比特币等区块链集成
这个符合原因3的目标，如果 IC 成为了新时代的互联网，那么自然需要和其他区块链接入来适应这个目标。
- 让更多的设备支持运行 IC
互联网之所以能广为传播，和多样化的设备支持脱不开关系。
- 将互联网原有生态迁移到 IC
为了实现原因3的目标，需要兼容互联网原有的生态，例如在 IC 上支持互联网流行的编程语言和编译执行环境。




02.

技术方向

Outline

- 02.1 技术方向介绍
- 02.2 WebAuthn
- 02.3 Actor
- 02.4 WebAssembly
- 02.5 容器化技术
- 02.6 小结

02.1 技术方向介绍



看到目录可能有大伙儿会问，这不是 IC 的讲座吗，为什么穿插了一系列的底层技术？

这里稍微解释一下，任何一种新技术的发展，都离不开既存的底层技术，就像搭积木，没有牢固的低层是没办法屹立滴。
😊

并且当你了解了一个项目使用的底层技术以后，对这个技术的知识还能用到其他同样使用该技术的项目上面。

介绍了底层技术的重要性以后，下面来谈谈 IC 各个模块中用到的底层技术。



02.2 WebAuthn

[WebAuthn](#) 是由 W3C 制定。IC 的 Internet Identity 项目主要用到这个技术。

WebAuthn 是利用[非对称加密](#)进行存储的，带 [HSM](#) 的设备和支持 [TPM](#) 芯片的计算设备都可以直接使用 WebAuthn。随着现代设备越来越多地引入 TPM 芯片和使用 HSM 技术，未来 WebAuthn 将会是无密码本地私钥存储的主流之一。

02.2 WebAuthn

- WebAuthn 的优点和缺点

IC 的[博客](#)上有一句话：

“The way that Internet Identity enables users to securely authenticate to dapps on the blockchain via their devices is revolutionary.”

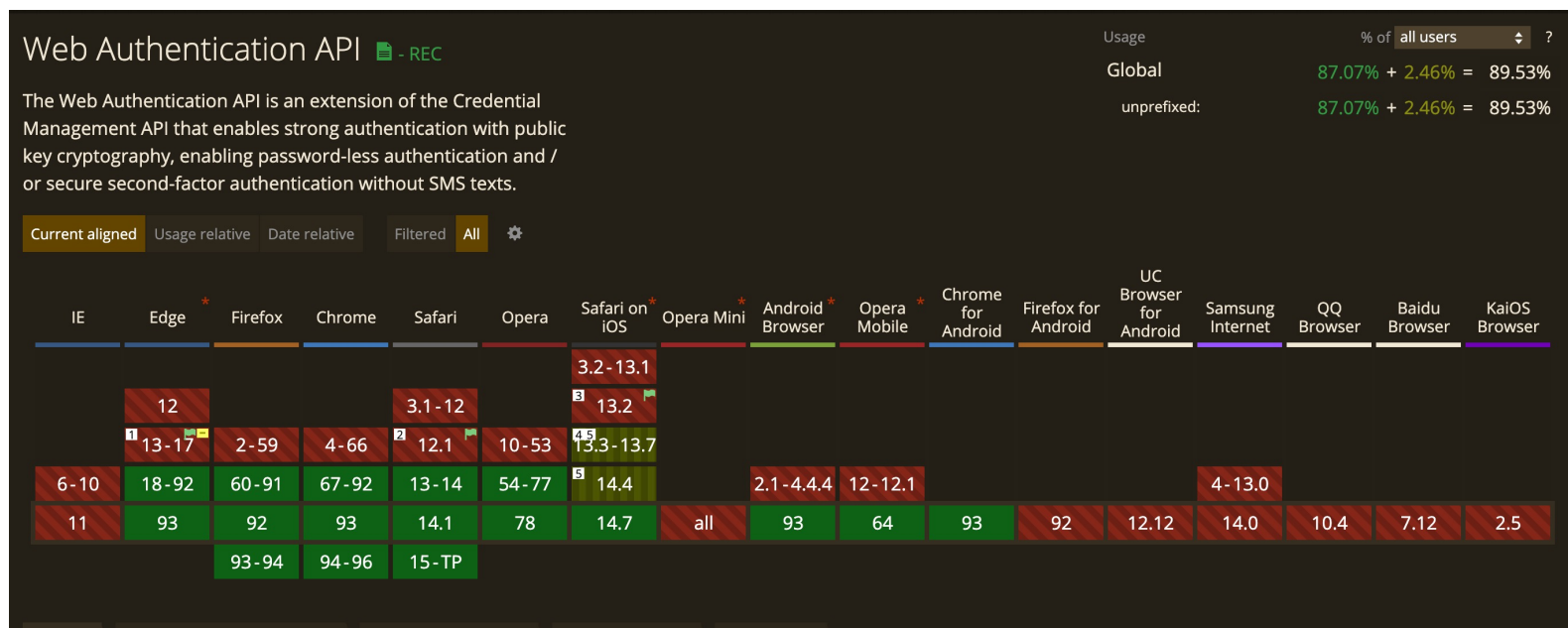
简单翻译一下，II 让用户通过设备在区块链上安全地认证到去中心化应用是革新的。

下面介绍一下 WebAuthn 的优点和缺点，从优缺点的分析可以看出，IC 在“革新”这方面的决心与挑战。

02.2 WebAuthn

I. 缺乏大范围的应用和支持

为了说明这一点，拿浏览器来举例。贴一张浏览器的支持情况，从 89.53% 的用户支持和浏览器支持可以看出，WebAuthn 离大范围的应用还有一些距离。☹



02.2 WebAuthn

II. 不支持较旧的设备

刚才提到了软件方面的限制，现在来谈谈硬件方面的硬伤。要用 WebAuthn，需要有带 HSM 或者 TPM 的安全模块的芯片的设备。一些较旧的手机或电脑，显然是没办法使用 WebAuthn 滴。😞

02.2 WebAuthn

III. 无需用户名和密码

这个优势之前已经聊过，WebAuthn 的一大优势是利用非对称加密进行存储和验证，通过设备或芯片读取私钥进行验证，省去了需要记住密码带来的麻烦。

IV. 匿名性

WebAuthn 在认证过程中使用基于椭圆曲线的 DAA 技术，可以达到保证用户隐私，防追踪的好处。

02.2 WebAuthn



其他类似的优缺点还有很多，在此不一一列举。下面来简单介绍一下 II（Internet Identity）在其上的扩展。

- Identity Anchor

没记错的话，这货以前叫 “Register”。简而言之它是一个设备-编号绑定的玩意儿（IA 的编号从 10000 开始，和腾讯 QQ 的起始号码一样）。Identity Anchor（以下简称 “IA”）有以下几种操作方式：

02.2 WebAuthn

1. 用正在使用的设备注册新的 IA

在这个步骤中，需要给这台设备指定一个名称，然后会让你选择加密验证的方式，较新的设备由于一般自带 HSM 或 TPM 安全模块，会有 “This Device” 这个选项。与之相对的，较老的没有安全模块的设备，则一般只有 “USB安全密钥” 这个选项，需要提供一个外部的支持 HSM 或 TPM 安全模块的 USB。注册成功后 II 系统会生成一个 IA 编号，之后可以拿该编号去和其他设备绑定，做到 “设备冗余性”。

“设备冗余性” 是防止设备出现意外丢失的一个策略。

02.2 WebAuthn

2. 用新的设备绑定已有的 IA

这个步骤除了将步骤1中输入设备名称操作替换为输入已有的 IA 外，其他操作（如验证方式）和步骤1一样。

3. 恢复无法访问的 IA

在步骤1中，注册成功后，会提供一个“seed phrase”，用这个“seed phrase”或者在注册时添加的“专用安全密钥”（洋文叫做“dedicated security key”）来恢复对某个 IA 的访问。

02.2 WebAuthn

- Pseudonym (匿名)

用某个 IA 访问 IC 上的应用程序时，II 会对每个将要访问的应用程序生成一个专用的密钥，IC 上的应用程序会验证这个密钥并让你以某个匿名身份来访问该应用程序。由于访问的每个应用程序都有单独的“专有密钥”，对于每个应用都有不同的身份，从而保证了应用程序之间的不可追踪和用户身份的匿名性。

02.2 WebAuthn

最后，II 由于是用智能合约编写的外部程序，调用 WebAuthn 的 API 接口与 IC 主体进行交互，属于 IC 上的官方“扩展类应用程序”，有关扩展类应用程序，可以参考 GitHub 上的[这篇文章](#)。

下面，我们来谈一谈 IC 智能合约中的 Actor 模型。

02.3 Actor

Actor，行话又叫 “[Actor model](#)”，是一种并行计算模型，属于计算的“基本单元”或“原语”；Actor 接收信息并进行计算；Actor 模型定义了一个计算单元或对象如何运作，以及与其他计算单元或对象的交互方式。IC 的智能合约（Motoko）设计中使用到了 Actor 模型。

- 使用 Actor 的优劣势

具体分析一下使用 Actor 设计智能合约有哪些好处和坏处，以及为什么 IC 选择使用 Actor 模型作为智能合约语言的设计模型。

02.3 Actor

I. 完全的隔离性

Actor 除了是最基本的“计算单元”外，每个 Actor 之间还具有完全的“隔离性”。举个例子，在 Actor A 上的操作无法影响到 Actor B 上处理的结果，对于处理的抗干扰性，是 Actor 之间实现完全隔离的必要条件。实现了完全的隔离性，也即实现了智能合约间互操作的安全性。

作为对比，Docker 虽然也能做到一定的隔离性，但是它不是完全隔离滴。即使两个容器分别在不同的网络，内存也不能做到完全的隔离。关于 Docker 内存的隔离性，具体可以参考[这里](#)。

02.3 Actor

II. 非并发的顺序执行

Actor 模型的一大劣势是无法并发执行，一个单独的 Actor 一次只能处理一个消息，如果需要并发执行，需要创建多个 Actor 来执行相对应数目的消息。

某些需要处理大量并发信息的“DeFi”应用，在考虑到资源消耗的情况下，再遇到 Actor 模型的时候，就不适用啦。

III. 无上限的扩容

由于 Actor 可以无限制地创建其他 Actor，这对于智能合约的扩展性方面得到了相当大程度上的提升。

02.3 Actor

IV. 异步

关于消息执行的异步，现代大多数系统已经实现了这一点。不再赘述，可以参照[这里](#)。

V. 高容错性

由于 Actor 可以通过消息来对其他 Actor 进行“操作”（如创建 Actor），Actor 之间同样可以获取状态，在一个 Actor 崩溃前，可以发送消息给另一个 Actor 来通知它进行容错处理（如重启）等。这在智能合约的可用性方面的提升又是一个里程碑。😊

02.3 Actor

VI. 多地域性/分布性

Actor 可以发布在不同的网络，并且支持跨域通信，一个 Actor 模型可以部署在多个不同的网络，即使一个 Actor 崩溃了，还有其他 Actor 可以用。进一步提升了智能合约的可用性。😊

有了上述几点，我们来分析一下 IC 为什么选择使用 Actor。

02.3 Actor

- 原因一: 建立安全的去中心化、可扩容的智能合约程序

记性还不错的同学，可能记得前面介绍过 DSS，IC 就是通过 Actor 模型拥有的几大优势（分布性、无限扩容性和完全隔离性）来实现 DSS。遗憾的是，由于 Actor 设计上的缺陷（无法并发执行），从而缺失了对智能合约的“效率”的考量。估计 Dfinity 早期设计 IC 的时候也没料到诸如 DeFi 这类应用会在未来成为主流趋势之一，否则为了兼容 DeFi 应用，也不会选择 Actor 这类不太适合 DeFi 使用的智能合约模型。

02.3 Actor

- 原因二: 和以太坊等其他智能合约公链竞争

像以太坊等智能合约公链都在语言或虚拟机等层面以直接或间接的形式实现了上述几点，IC 作为新一代公链，自然需要寻找一门适合在公链领域使用的智能合约设计模型来和其他公链作竞争，于是 Actor 模型就入选了。😊

介绍完了 Actor 模型，下面再来介绍一下 Mokoto 语言最终执行的形式 “WebAssembly”。

02.4 WebAssembly

[WebAssembly](#)（以下简称 Wasm）由 W3C 发起，Mozilla、Microsoft、Google、Apple 等企业跟进。

这里插一句，之前提到的 WebAuthn 项目也是以 W3C 为主发起的。从这里可以看出，IC 项目团队在做项目规划时很多思路都来源于 W3C。

Wasm 是用来定义与规范 Web 的标准，是新一代的 Web 范式，它包括但不限于用于 Web 应用程序的交互。目前，C/C++、C#、Rust 等语言都能通过 API 接口编译成 Wasm。

扯的有点多，下面来介绍一下 Wasm 和 IC 团队的关系，以及为什么 IC 团队会想到用 Wasm。

02.4 WebAssembly

- Wasm 和 IC 团队的关系

可能有同学会问，为啥一开始说这个捏？要介绍 Motoko 这门语言的来由，需要先介绍一下 [Andreas Rossberg](#) 其人。

Andreas Rossberg 作为 Wasm 的作者之一，在 Wasm 和程序语言的设计上有相当的经验，请这位大神来设计 Motoko 这门语言和指导 Wasm 相关的开发，或许是再合适不过。

也许是 IC 团队看到 Andreas Rossberg 在这方面的实力，自己团队刚好又缺有语言设计相关经验的人，于是就把 AR 从 Google 请来了。😊

02.4 WebAssembly

- 为什么选择 Wasm 作为 IC/Motoko 的主要执行模块？

上面说了结论，下面来分析一下原因。

话说 IC 的原生语言（Motoko）和使用语言（Rust）都需要编译成 Wasm 二进制，然后才能在 IC 上运行。为什么需要费尽周折再编译而不用语言自带的二进制执行模块呢？这就要从 Wasm 的优势和 IC 团队设计上的考量两个角度来进行分析。

02.4 WebAssembly

- Wasm 的优势

I. 开源、跨平台

这个不用多说，开源的好处是可以容纳更多的开发者社区，从而使更多语言支持编译成 Wasm。而借助跨平台的优势，可以在多个平台上执行 Wasm 二进制程序。

II. 原生级别的执行效率

Wasm 通过利用[通用硬件能力](#)，定义了一套硬件标准，只要某个平台符合这些标准，就能够达到该平台原生的执行效率。

02.4 WebAssembly

III. 隔离的执行环境

Wasm 从设计上的执行环境是完全隔离的，由于环境的隔离，程序执行过程中占用的内存是不共享的，程序之间不共享内存，从一定程度上提升了执行的安全性。

关于 Wasm 的优势还有很多，例如互联网原生（直接嵌入在 Javascript VM），在此不一一赘述，下面来聊一下 IC 团队在设计上的考量。

02.4 WebAssembly

- IC 团队设计上的考量

IC 团队在设计 Motoko 语言的时候，提到了几点用 Wasm 的原因。具体可以参考这篇[答疑](#)和[博客](#)，这里就不再复述了。

这里说一下我个人的理解，除了语言不敏感、跨平台兼容性、Web 原生、速率和安全外，一个很重要的原因是 Wasm 和 Actor 在执行环境上的“兼容”。

02.4 WebAssembly

前面提到，Actor 在设计上具有类似沙盒的内存隔离，这对于智能合约来说是一个合适的设计。而 Wasm 在执行环境上的隔离，可以确保运行在 IC 上的所有程序在静态和动态的状态下能够做到完全隔离。

什么意思捏？举个例子，就是程序运行和非运行时的情况，或者读和写的情况，在这些情况下，程序之间可以做到相互隔离。“相互隔离”这一点，对于一组不完全信任的节点，或者说公链，是非常有用滴。

对于其他语言（Rust 等），IC 则是设计了 [Candid](#) 这套接口定义语言来与 Actor 的消息机制作兼容。

02.5 容器化技术

最后来聊一下容器化技术。有了上述两个模块（Actor 和 Wasm），实现 IC 上的容器化（[Canister](#)）就容易很多了。

首先，Wasm 模块负责容器状态的存储，每个 Wasm 程序可以提供 4GB 的存储空间。

其次，Actor 模块定义消息的类型，消息可以分为读取类型和写入类型，读取类型的消息读取 Wasm 中的状态，而写入类型的消息用来执行 Wasm。

可能大多数同学都注意到了，对比如今的 Docker 来说，使用这类容器技术，大大减轻了系统的负载（不需要再安装操作系统），而且由于 Wasm 的特性，实现了应用原生的执行效率。在未来，可能会成为“应用程序容器化”的趋势。



02.6 小结

最后做个总结。

在上述提到的技术方案中，Wasm 和 WebAuthn 是由 W3C 为主发起的，属于比较上层的技术（只需要调 API 或者集成就能用）；而 Actor 和容器化技术是一种设计理念，属于比较底层的技术，需要用 Actor 来设计语言，容器化技术来设计 Container，而是不能直接拿来用滴。



03.

代表性项目

Outline

- 03.1 官方的代表性项目
- 03.2 第三方的代表性项目

03.1 官方的代表性项目

官方的代表性项目，也就是由官方开发，在 IC 上运行的 Dapp，主要有以下几类：

- 功能性项目

功能性项目的代表主要有：[Internet Identity](#)、[Rosetta API](#)、[Cycles Wallet](#)。

- 工具类项目

关于工具类项目，可以参考我在 GitHub 上写的[官方的扩展类应用](#)，所有命令行类的项目都属于工具类项目。

03.1 官方的代表性项目

- 其他项目

这类应用有很多，大多数是以某个 Use Case 来做的项目，可以参考[官方的扩展类应用](#)中的应用程序类项目。

03.2 第三方的代表性项目

非官方的项目，都属于“第三方”项目。可以分为如下几种：

- 功能类

功能类的项目，如区块链浏览器 [ic.rocks](#) 和托管服务 [Fleek](#)。

- 工具类

工具类项目，如 Fleek 的 [IC Deploy Action](#)、chenyan 的 [ic-repl](#) 和 [sudograph](#)。

- 其他

根据 Use Case 来做的项目都划分到这类。目前这类比较火的项目主要有：[ic-wall](#)、[DSCVR](#) 等。



04.

展望

04. 展望

在概要 01.3 中，我们从第一人称视角对 IC 的发展趋势作了分析和展望。

下面从第三人称视角来谈谈 IC 今后的发展，在 IC 未来的发展中，会不可避免和如今的互联网巨头（[GAFAM](#)）竞争。作为新一代的分布式云计算平台，IC 需要在以下几点做出努力：

- 吸引更多的开发者社群

互联网、互联网巨头的早期离不开开发者社群的支持，要吸引更多开发者参与进来，所定义的开发范式、通用性以及较低的开发成本是关键。

04. 展望

- 更友好的用户体验

从近期的 [4GB 扩容案](#)来看，要做到更好的用户体验，其中一个就是要提升应用的执行效率，应用执行效率提高了，应用的速度变快，用户自然也有了。应用的体验差，区块链的名头再好也是没有用滴。

- 先进的技术

除了用户体验和开发者社群，最重要的还是技术本身。前面的分析提到过，IC 借鉴了几项 W3C 研发的目前来说还算领先的技术。在站在巨人肩膀上的同时，自己的团队内部也需要提出一些创新的方案。如今的互联网巨头，最开始也是靠着某一个特殊的创新点而脱颖而出的。IC 如果需要发展，也必须从内部开始创新。

THANKS

谢谢观看

时间：2021.09.22

