# 护城河为什么会一夜蒸发

# 导读

看懂决定软件公司几十年兴衰的那几股力量

先讲三个死法，都不体面。

网景（Netscape：1990 年代最火的网页浏览器公司）巅峰时占据了九成市场，浏览器几乎是它的同义词。几年后它没了。柯达式的没落？不，它产品不烂、用户上亿、账上也有钱。诺基亚卖手机卖到全球第一，转身就成了博物馆展品。曾经统治数据库的公司、统治办公软件的公司，一个个从"你绕不开我"变成"谁还用它"。

这里有个反直觉的事实：**杀死它们的，往往不是没钱、不是产品变烂、也不是用户跑光。**出事的时候，钱、产品、用户常常都还在。它们输给的是一种看不见的东西——你可以叫它护城河（moat：让对手打不进来、让你能长期赚钱的那道屏障）——护城河还在的时候你感觉不到它，等它蒸发了，城墙一夜垮塌，你才知道原来一直靠它撑着。

这本书想搞清楚的，就是这道河为什么会一夜蒸发。

## 它不是一本历史书

我知道你可能担心什么：又一本讲盖茨、讲乔布斯、讲硅谷八卦的编年史。不是。

历史在这本书里只是**论据**，不是主角。IBM、微软、Oracle、谷歌、开源运动、Salesforce、AWS、App Store、OpenAI——它们会一个个出场，但我不关心谁在哪年融了多少钱、谁跟谁反目。我关心的是它们身上反复出现的**那几股力量**：

- 软件几乎不要成本就能复制一份（零边际成本：多卖一份的额外花费近乎为零）——这既能造出躺赚的垄断，也让你天生脆弱，因为对手复制起来同样便宜。
- 谁站在平台上，谁就是平台——被别人当地基踩，和把别人踩在脚下，是完全不同的命。
- 转换成本和绑定：让你走不掉，比让你满意更值钱。
- 网络效应、把"免费"当武器、开源掀桌子、数据越用越聪明的飞轮……
- 还有一股最沉的地心引力：软件挣钱的方式，几十年里从卖授权，滑向广告、滑向订阅、滑向云——顺着滑的公司活了，逆着站的公司被拽下去了。

同样几股力，在不同年代、不同公司身上一遍遍重演。看懂了力，你就不用记那么多故事了。

## 你能带走什么

一副眼镜。

物理里有个动作叫受力分析：不管东西多复杂，先把所有作用在它身上的力一根根画出来，再看它往哪动。这本书想给你的，就是一副能对任何软件公司做受力分析的眼镜。

> 大白话
>
> 戴上它，你看一家公司就不再只看它今天多风光、赚多少钱，而会下意识问：它靠哪几股力撑着？哪股力正在松动？如果某根柱子抽掉，它还站得住吗？

这副眼镜不挑行业。软件是它最锋利的试验场，因为软件里这些力量走得最快、最极端——别的生意几十年才演完的兴衰，软件几年就演一遍。但你会发现，同一副眼镜拿去看任何一门生意，一样好使。

> 能长盛几十年的公司，和巅峰即坠的公司，用的是同一套物理。差别只在：它们各自被哪几股力抓住了。

好，别急着记名字和年份。翻开第一章，我们从软件这门生意最古怪的一条性质说起——它几乎不要钱就能复制，而这一条，几乎解释了后面所有的荣枯。

# 第一章 · 卖的到底是什么

你有没有想过一件怪事：造一辆车，第一辆和第一百万辆，每一辆都得实打实地耗掉钢铁、橡胶、几个小时的装配工时。可造一份软件——比如你正在用的操作系统——第一份可能烧掉几百个工程师几年的时间和几千万美元，但复制出第二份、第一百万份，成本几乎是**零**。按个键，它就在那了。

这不是一个无聊的细节。这本书要回答的核心问题是：为什么有的软件公司能称霸几十年，有的却巅峰即坠？答案里反复出现的第一股力量，就藏在上面这件怪事里。我们先把这股力量看清楚，因为后面所有的故事——平台、锁定、网络效应、免费绞杀——全都从它长出来。

## 软件是一种"反常物理"的东西

经济学里有个词叫边际成本，意思很简单：你已经在生产了，再多做一个，得额外掏多少钱。多烤一个面包，得多一份面粉和电费——面包的边际成本不低。多印一本书，得多一份纸和油墨——也不是零。

软件的边际成本近乎为零。写好之后，多给一个用户用，你几乎不用多花一分钱。这个"近乎为零"听起来像个技术小事，但它一旦成立，就会像一条物理定律一样，逼着整个行业往一个方向走。

> 大白话
>
> 说人话：造实物的世界里，你卖得越多，成本堆得越高，所以市场能容下很多家厂商，谁也吃不下全部。软件的世界里，你卖得越多，摊到每份上的成本越薄，第一名越滚越大，第二名根本追不上。物理法则变了，结局自然就变了。

## 零边际成本，为什么会"天生走向垄断"

想象两家做同类软件的公司，A 有一千万用户，B 有一百万。他们当初的开发成本假设都是一亿美元。A 把这一亿摊到一千万人头上，每人身上只压了 10 块；B 摊到一百万人头上，

每人压了 100 块。

这意味着 A 可以卖得比 B 便宜十倍还赚钱，或者卖一样的价钱、拿多得多的利润去砸研发、砸市场、挖人。B 越落后，摊销的包袱越重，越追不上。这不是谁更聪明，是数学在推着走：**在零边际成本的行业里，领先本身会制造更多领先。**大者恒大，赢家通吃，垄断不是意外，是默认结局。

做实物的行业没有这种碾压。钢厂再大，多炼一吨钢还是得买那么多铁矿石，规模优势有个天花板。软件没有这个天花板——多一个用户几乎白送，所以头部公司能一路滚到吃下大半个市场。

## 同一枚硬币的另一面：天生脆弱

但这里有个反直觉的转折，也是这本书真正想让你记住的：**让软件走向垄断的那个特性，同时也让它脆弱得吓人。**

为什么？因为一份软件的全部价值，都在"人们愿意用它"这件事上，而不在任何实物里。一座炼油厂就算生意黄了，厂房、管道、储罐还值一大笔钱，能变卖、能改做别的。一个统治市场的软件一旦被更好的东西替代，用户几周内就能装上新的、删掉旧的——它剩下的资产几乎是零。没有厂房可卖，没有库存可清，护城河干涸的那一刻，什么都不剩。

这就是本书标题里"一夜蒸发"的物理根源。实物资产是慢慢折旧的，软件的价值是**悬在用户选择上的**——支撑它的是人的习惯，不是钢筋水泥。习惯断了，价值就断崖。你会在后面几章里看到 Lotus 1-2-3、WordPerfect、Netscape 这些当年的霸主，怎么在很短的时间里从"人人都用"跌到"没人再提"。它们不是被造得更好的产品慢慢磨死的，是被替代之后一无所有。

大白话

记住这对双胞胎：零边际成本让软件公司容易做到无比大，也让它们输起来无比快。同一个特性，既是它的加速器，也是它的悬崖。这本书讲的很多兴衰，本质都是这两股劲在拔河。

## 软件曾经不是拿来"卖"的

奇怪的是，这门天生能造垄断的生意，一开始根本不作为独立商品存在。要看清"卖软件"这件事本身是怎么诞生的，我们得回到 1969 年，回到 IBM——当年那家几乎定义了"计算机"三个字的公司。

那时候你买一台 IBM 大型机，软件是随机器白送的。厂商的算盘很清楚：软件是硬件的赠品，是帮你卖出那台昂贵铁疙瘩的诱饵。IBM 甚至会把源代码给你，附近还有一大群独立小公司靠给这些机器写程序糊口。软件本身没标价——它像买车送的脚垫，附属品而已。

1969 年 6 月，IBM 宣布了一件后来被叫作 软件解绑（unbundling，就是把原来"绑在一起卖"的东西拆开、各自单独标价）的事：从此软件、服务和硬件分开定价，软件要单独收费了。

为什么要拆？一部分是外部压力——美国政府正在对 IBM 展开反垄断调查，"白送软件"被视为一种挤垮小对手的手段，因为没人竞争得过"免费"。IBM 主动拆开，是为了减轻反垄断的火力。历史的一个反复出现的模式在这里第一次露头：**免费捆绑，是巨头碾压对手最锋利的武器之一。**这把武器我们留到最后一章，看微软如何用免费的浏览器把 Netscape 逼死。

## 拆开的那一刀，划出了一整个行业

解绑真正的历史分量，不在反垄断本身，而在它无意间干成的一件大事：**它让"软件"第一次成了一件可以单独标价、单独买卖的商品。**

一旦软件能独立卖钱，一个此前不可能存在的算盘就成立了：我不造硬件、不卖机器，我只写一份程序，然后卖给成千上万台机器上的成千上万个用户。前面那条"反常物理"——一次开发、无限复制、边际成本近乎为零——从这一刻起，才第一次有机会变成一门真正的生意。独立软件公司，作为一个物种，从这里开始批量出现。

你可以把 1969 年这一刀，理解成给前面那台"垄断-脆弱"引擎第一次点火。引擎的原理（零边际成本）本来就在，但只有当软件被允许单独卖，这台引擎才开始真正运转，才开始一次又一次地造出巨头、又一次又一次地让巨头一夜蒸发。之后半个世纪软件业所有的惊心动魄，都是这台引擎在不同场景下的重演。

## 这一章你该带走的东西

把最硬的那颗钉子钉牢：

- **软件的边际成本近乎为零**——多一个用户几乎白送。这是它区别于一切实物生意的根本。
- 正因如此，它**天生走向垄断**：领先会自我放大，赢家通吃是默认结局，不是运气。
- 也正因如此，它**天生脆弱**：价值全悬在"人们愿意用"上，没有实物兜底，被替代就一无所有、可能一夜蒸发。
- 而"卖软件"这门生意本身，是 1969 年 IBM 解绑那一刀划出来的——引擎的原理早在，那一刀才点了火。

但这里还漏了一个大问题。就算你写出了一份能称霸的软件，它也不是凭空运行的——它得跑在某台机器、某个系统上。那个"地基"是谁的？如果地基是别人家的，别人随时能把它抽走，或者自己下场做你这一块，那你的垄断和脆弱，就还要再叠上一层别人的意志。下一章，我们就去看那些站在别人地基上的公司，是怎么在地基被换的那一瞬间轰然倒下的。

# 第二章 · 站在谁的地基上

先讲个让人后背发凉的事实：这几家公司死的时候，账上都不缺钱，产品也不烂，用户还多得很。

Lotus 1-2-3 一度是 PC 上最好的表格软件，好到很多人买电脑就是为了跑它，市占率碾压全场。WordPerfect 是当年文字处理的王者，律师所、政府机关整栋楼都在用它写文件。它们不是被某个更聪明的创业公司偷袭死的——它们是在自己巅峰的那几年里，眼睁睁看着脚下的地基被人抽走，然后集体坠落。

这一章要讲的，就是那股抽地基的力量：**你的软件不是长在真空里的，它长在别人的地基上；而那个地主，随时可以涨你的租、把你隔壁的黄金铺位租给你的对手，甚至自己下场开一家一模一样的店，正好挡在你招牌前面。**

## 什么叫"站在别人的地基上"

先把两个词说清楚，别让它们裸奔。

操作系统，就是管着整台电脑最底层的那套软件——它决定了屏幕怎么画、文件怎么存、键盘敲下去信号传到哪。你写的任何一个程序，最终都得靠它帮你把活儿落到硬件上。你不直接跟电脑说话，你跟操作系统说话，操作系统再跟电脑说话。

平台，说白了就是"别人在你这上面盖房子的地基"。操作系统是最典型的平台：无数软件公司在它上面盖自己的房子（写自己的程序），卖给用户。地基是谁的？地主说了算。

大白话

你开了一家很赚钱的餐厅，但店面是租来的商场铺位。商场老板手里有三张牌能随时打你：一是涨租，二是把你正对面的黄金铺位租给一家跟你卖一模一样菜的对手，三是干脆自己在商场正门口开一家同款餐厅，正好挡住通往你店的路。你菜做得再好，也架不住这三张牌里的任何一张。软件公司跟平台的关系，就是这个不对称：你离不开它，它随时能换掉你。

## 微软是怎么当上地主的

故事得从 1981 年说起。IBM 要做个人电脑，急着找一套操作系统，找到了当时还很小的微软。微软手里其实没有现成的东西，转手买了一套别人的系统改吧改吧，就成了 MS-DOS（微软的磁盘操作系统，早期 PC 上那套黑底白字、要打命令的底层软件）。

这里有一步棋，几乎决定了后面二十年的格局：微软没有把 DOS 一次性卖断给 IBM，而是保留了向别的电脑厂商授权同一套系统的权利。

这意味着什么？很快市面上冒出一堆"兼容机"——不是 IBM 造的，但能跑一模一样的 DOS、一模一样的软件，还更便宜。每卖出一台兼容机，微软就收一份授权费。硬件厂商打成一片、利润薄如刀片，而躺在所有这些电脑最底层、每台都要过路费的，是微软那一套系统。

**地主不生产房子，地主只收租——但天下所有的房子都盖在他的地上。**到 Windows 出来的时候，这个地基已经铺满了全世界的桌面。

## 第一张牌：把地基换了，你就得从头再来

八十年代末，微软要从命令行的 DOS 换成有窗口、能用鼠标点的 Windows（图形界面操作系统，就是我们今天还在用的那种桌面）。这是一次地基的大搬家。

对 Lotus 来说，这本该是一次普通的"跟着升级"。但 Lotus 押错了宝——它赌的是另一套叫 OS/2 的系统会赢（IBM 和微软早年合作、后来分道扬镳的产物），于是把主要力气投在了那边，Windows 版的 1-2-3 迟迟做不出来，做出来的也不够顺。

而微软自己的表格软件 Excel，早就在 Windows 上准备好了——毕竟地基是它自己铺的，它比谁都清楚新地基长什么样、什么时候铺好。地基一换，用户涌向 Windows，桌面上第一个能用得顺手的表格软件是 Excel，不是 1-2-3。等 Lotus 的 Windows 版姗姗来迟，好位置已经被占完了。

WordPerfect 死得几乎是同一个剧本：DOS 时代它是文字处理之王，但 Windows 版又晚又难用，而微软的 Word 早在新地基上等着。地基换代的那一两年，就是王座易主的那一两年。

注意，这里没有一家公司"犯蠢"。他们都是当时最能干的团队，只是站在了一个结构性必输的局里——地主要搬家，房客总是慢半拍，而地主自己搬起来零成本。

## 为什么聪明公司会集体踩坑：三个"差"

把上面的故事拆开看，你会发现失败不是因为谁笨，而是因为房客和地主之间天然存在三种不对称。我把它们叫做三个"差"。

- **信息差**：新地基长什么样、哪些接口会变、什么时候正式发布，地主自己最先知道，而且知道得最全。房客只能等文档、等测试版，永远比地主晚一拍看清全貌。
- **时间差**：等你终于把房子在新地基上重盖好，地主自家的同款房子早就盖完开门营业了——它的团队几乎是零迁移成本，因为盖房子的人和铺地基的人坐在一栋楼里。
- **动机差**：地主换地基，从来不是为了照顾你。它换是为了自己那盘更大的棋（比如推自己的 Office）。你的兴衰在它的账本上，最多是个小数点后的数字。

这三个差叠在一起，就把"某家公司管理失误"这种解释推翻了。不是谁失误，是**只要你的命根子架在别人的平台上，这个必输的结构就一直在那儿等着**——今天没触发，只是地主还没决定搬家而已。

## 第三张牌有个专门的黑话

地主亲自下场、开一家同款店挡在你门口——这张牌在软件圈里常见到有了专门的说法：被 sherlocked，意思是"平台方把你这个第三方产品的功能直接吸进自己系统里，让你瞬间失去存在的理由"。

这个词的来历得说准，因为它很容易讲反。苹果自家早就有个系统内置的搜索工具，叫 Sherlock。后来有家小公司 Karelia 做了个更好用的第三方工具，叫 Watson。结果 2002 年，苹果发布新版 Sherlock，把 Watson 的功能几乎照搬进了自家系统——用户已经有 Sherlock 了，谁还去单独装 Watson？Watson 就此死掉。**是自家的 Sherlock 吃掉了第三方的 Watson**，所以后来大家用"被 sherlocked"形容"被平台方亲手做进系统而挤死"这件事。

Lotus 和 WordPerfect 遇到的，其实就是同一张牌的更大号版本：微软不是吃掉一个小工具，而是把整个品类——表格、文字处理——都做进了自己主导的平台里。

## 这一章，只讲这一股力

你可能会想：那用户为什么不干脆坚持用 1-2-3、用 WordPerfect？换个软件不就行了吗？——这里其实藏着另一股力，叫**转换成本**，它是下一章的主角，这里先不展开。

你也可能想：Windows 后来那么难被推翻，是不是因为"用的人越多就越离不开"？——那叫网络效应，第四章专门讲。至于微软后来怎么用"免费送"这一招把对手直接送走，是第五章的事。

这一章只钉死一件事：**软件天生要站在某个平台上，而平台方和你之间的关系是根本不对称的。**地基是它的，牌在它手里，你能不能活过下一次地基搬家，很大程度上不取决于你有多能干，而取决于你有没有意识到——你从一开始，就站在别人的地上。

# 第三章 · 锁住你的不是产品

上一章我们看到，一家软件公司再厉害，只要它的地基是别人的平台，平台方一翻脸，它就可能连夜坍塌——Lotus、WordPerfect 就是这么没的。但这里有个绕不过去的疑问：**为什么有些软件明明被人骂了几十年、明明有更好更便宜的替代品，用户却还是走不掉？**

你身边一定有这样的人：一边骂 Excel 难用、骂 Oracle 贵得离谱，一边继续年复一年地交钱。这不是斯德哥尔摩综合症，这是本章要拆的那台机器——转换成本，也就是"你想换掉它，得付出的全部代价"。这个代价不只是新软件的钱，而是你迁移数据、重学操作、重新培训全公司、承担出错风险的总和。当这个总和高到让你宁可继续忍着，护城河就成了。

## 产品好不好，根本不是重点

先纠正一个直觉。大多数人以为，软件公司靠的是产品做得好，别人做得差，所以赢。这话对了一半，但漏掉了最关键的一半。

真正长寿的软件公司，护城河往往不在产品本身，而在**用户为了用它，顺手往里塞进去的一切东西**。你在 Excel 里攒了十年的报表、公式、宏；你公司三千个员工的手指记住了每个菜单在哪；你所有的合作方发来的文件都是 `.xlsx`。这些东西，微软一分钱没花，是你自己免费帮它砌进护城河里的砖。

> 大白话
>
> 说人话：产品是钩子，把你钩住的却是你自己往里投入的时间、数据和习惯。软件公司最聪明的地方，是让"离开的痛"由你来承担，而不是它来制造。它只要在最开始把钩子递给你就行。

所以判断一家软件公司能不能活几十年，别光看它今天产品有多惊艳，要问一句：**用户想走的时候，得先扔掉多少自己的东西？**扔得越多、越舍不得，这家公司就越安全。

## Office 的杀招：把"文件"变成地雷

微软 Office 是把转换成本玩到极致的教科书。但它真正的杀招，不是 Word 排版多好、Excel 算得多快，而是文件格式——就是那个决定你的文档"长什么样、怎么存"的内部编码

规则。

早年的 `.doc`、`.xls` 格式，是不公开的、私有的、只有微软自己完全搞得懂的一套编码。这一手的狠处在于：格式一旦变成事实标准，它就不再是"微软的产品"，而是"全世界的共同语言"。你写一份合同用 Word，因为收件方要用 Word 打开；收件方用 Word，因为发件方都发 `.doc`。每个人锁住的，其实是每一个别人。

这就有意思了——你会发现光是"我自己想换"根本没用。哪怕你个人受得了别的办公软件，你也换不动，因为你得先说服跟你来往的所有人一起换。这道墙不是微软一家砌的，是全世界几亿用户互相砌起来的。这里其实已经悄悄踩到了下一章要讲的网络效应（用的人越多、这东西对每个人就越值钱），本章你先记住一点：**转换成本可以是一个人的，也可以是一整个群体互相绑死的——后者几乎拆不开。**

后来有人问：那为什么开源的 OpenDocument（一套公开的、谁都能读写的办公文档格式标准）没能翻盘？技术上它完全够用，还免费。答案就在这台机器里：技术够用，从来不等于换得动。你把几百万份历史文档、几十年的公式、全公司的操作习惯全押在了旧格式上，一个"技术上更好"的新格式，撬不动这堆沉没成本。**更好，赢不了更难离开。**

## Oracle：把锁做进数据库的地基里

如果说 Office 锁的是"人和文件"，那 Oracle（做企业级数据库起家的公司，数据库就是企业存放核心业务数据的那个大仓库）锁的东西更深——它锁的是一家公司的命根子：所有客户、订单、财务、库存数据，全存在它那里。

数据库这门生意的转换成本，是所有软件里最高的一档，原因有三层，一层比一层狠：

- **数据搬不动。**把上亿条业务记录从一个数据库倒进另一个，不是复制粘贴，是一场可能持续一两年、稍有闪失就丢数据的大手术。而这些数据一旦丢了，公司可能就没了。
- **代码焊死了。**企业为了榨干性能，会大量用 Oracle 特有的功能和它自家的编程方言 PL/SQL（Oracle 专属的数据库编程语言，别家不认）。这等于把业务逻辑用只有 Oracle 听得懂的话写死了，换数据库就得把这些全部重写。
- **没人敢担责。**核心数据库一动，整个公司业务停摆的风险就压在拍板的人头上。于是"别动它"成了最安全的选择——这是一种心理上的转换成本，往往比技术上的还硬。

Oracle 太清楚这三层锁的价值了，所以它的定价和销售策略出了名的强硬：一旦你进来，续费涨价、审计合规、追加授权，你几乎只能接受。因为它知道，你算过那笔账——迁走的代价，比忍受涨价高得多。这就是为什么 Oracle 能靠一批几十年前就锁进来的大客户，稳稳收几十年的钱。**它卖的从来不是数据库本身，而是"你离不开"这件事。**

> 大白话
>
> 说人话：Oracle 最值钱的资产，不是它的技术，是它客户名单上那些"想走走不了"的公司。技术会被追平，但那笔"迁移要花一两年、还可能出人命关天的事故"的账，几十年都不会变小。

## 护城河的另一面：它也会反噬

但你要是以为转换成本是万灵药，那就把这本书读浅了。这道护城河有个阴暗面：**它锁住客户的同时，也在锁住公司自己。**

为了不破坏那份宝贵的"离不开"，被锁住的一方是客户，主动权在公司手里，看起来公司稳赢。可代价是：公司会变得害怕改变。微软为了兼容那堆旧 `.doc` 文件，背了二十多年的历史包袱，想大改都改不动；Oracle 为了不吓跑老客户，很多老设计一直不敢动。**你用来锁别人的那些东西，最后也会反过来锁住你自己的手脚。**

更要命的是，转换成本再高，也高不过一次**范式的整体搬家**。当整个行业从"装在自己机房的软件"搬到"租用别人云端的服务"，客户本来就要推倒重来、从头搭一套——这时候，那道靠"搬家太麻烦"砌起来的墙，一夜之间就没用了，因为客户反正都要搬一次家了。护城河没有蒸发，是脚下那块地整个被换掉了。这也是为什么护城河最深的公司，在大转型面前反而最迟钝——它们太习惯"没人走得掉"了。

所以转换成本是真护城河，但它防的是"用户一个个溜走"，防不住"整个世界换赛道"。到这里我们已经见过两种力量：平台能抽走你的地基，绑定能锁住你的用户。可还有一股力量，比它们都更凶——它不是让你难离开，而是让离开你的人自己吃亏。**用的人越多，这东西就越值钱，值钱到后来者根本挤不进来。**下一章，我们来拆这台会赢家通吃的机器。

# 第四章 · 越多人用越离不开

上一章我们看到，微软用文件格式把用户锁在了 Office 里——你不是不想走，是走的代价太高。锁定是"你离开很疼"。这一章讲一股更狠的力：它让你根本**不想**走，因为别人都在这儿。而且越晚来的对手越没戏，不是他们东西差，是他们生错了时候。

先抛一个反直觉的现象：一个软件，功能一模一样，甚至更烂，却能把一个技术上明显更好的对手活活饿死。凭什么？答案不在产品里，在**产品之外的人**。

## 为什么"多一个人用"能改变价值本身

普通商品你买一件，值多少就是多少，跟别人买不买没关系。你买双跑鞋，隔壁一万个人也买同款，你这双不会因此变好穿。

但有一类东西不一样。网络效应——说人话就是：这东西的价值，取决于有多少人跟你用同一个。你手上这份软件值不值钱，一半看它本身，另一半看"还有谁在用"。

电话是最老实的例子。世界上只有一部电话，它值零，因为你打给谁？两部电话，能打一通。这里藏着一个要命的算术：**价值不是随人数一根根加上去的，是随人数成对地涨上去的**。

> 大白话
>
> 10 个人的网络，能连出多少对通话？大概 45 对。100 个人呢？将近 5000 对。人数涨了 10 倍，"能连出来的关系"涨了 100 倍。这就是"超线性"：投入翻倍，回报翻好几倍，而不是也翻一倍。工程师熟悉这个——它就是 $O(n^2)$。用户数是 n，价值大致跟着 $n^2$ 跑。

这条 $n^2$ 曲线是本章所有故事的引擎。它一旦转起来，会干出三件让局外人看不懂的事：一是**赢家通吃**，第二名不是小一号的赢家，而是直接归零；二是**后来者几乎无法翻盘**，哪怕产品更好；三是**转折点极其突然**——某天之前谁都能赢，某天之后大局已定，中间没有缓冲。所谓"护城河一夜蒸发"，很多时候就是这条曲线在反方向上给别人助攻。

## Windows：一张床上睡了两拨人

Windows 的真正护城河，从来不是那个开始菜单。是它把两拨互相需要的人，绑上了同一条 $n^2$ 曲线。这叫双边网络——一个东西同时服务两群人，这群人越多，那群人越离不开，反过来也一样，两边互相拽着往上冲。

这两拨人是：**用户**和**开发者**（写软件的人）。它们之间有一个死循环，方向对你就是飞轮，方向反你就是绞索：

- 用 Windows 的人多 → 写软件的人当然优先给 Windows 写，因为客户都在这儿；
- 能在 Windows 上跑的软件多 → 新用户当然买 Windows，因为想用的东西都在这儿；
- 于是用户更多 → 开发者更多 → 软件更多 → 用户更多……

注意这里有个关键点：**用户不是在为操作系统买单，是在为"那几千款只有 Windows 上才有的软件"买单**。操作系统本身谁在乎呢，人们要的是能报税、能画图、能打的那个游戏。到上世纪九十年代，个人电脑操作系统里 Windows 占了九成以上——注意不是"好一点点"的领先，是碾压式的独占。

这时候你要做一个新系统来抢，会撞上一堵物理学都帮不了你的墙。你的系统技术上再优雅，开发者也不给你写软件——因为你没用户；用户也不来——因为你没软件。你被卡在循环的最底部，两边互相等对方先动，谁都不动。这堵墙有个名字叫鸡生蛋问题：要用户得先有软件，要软件得先有用户，你一个都拿不出来。

历史反复验证这一点。IBM 亲自下场做过 `OS/2`，技术上被不少工程师认为比当时的 Windows 更扎实、更稳——结果呢？没人给它写足够多的软件，用户没东西可用，慢慢就荒了。**把"更好的产品输给更差的产品"这件事从意外变成规律的，就是网络效应。**产品好坏是加分项，网络大小是命门。

## 为什么"更好"救不了后来者

工程师最难接受的就是这一条，所以掰开讲。假设你做了个新系统，各方面比在位者好 20%。听起来稳赢？我们把它放回 $n^2$ 曲线上看。

在位者有一亿用户，它的网络价值大致正比于一亿的平方。你产品好 20%，可你只有一万个种子用户，你的网络价值正比于一万的平方。一亿的平方比一万的平方，差了**一亿倍**。你

那 20% 的产品优势，在一亿倍的网络差距面前，连个零头都算不上。用户看的是"我能连上多少人、能用多少软件"，不是你代码写得多干净。

大白话

所以后来者要赢，靠"做得更好一点"是没用的，那点好被 $n^2$ 直接吃掉了。真正能翻盘的路子只有两条：要么找一块在位者的网络够不着的角落，在那儿从零单独长出自己的小网络（比如先占死某个在位者看不上的细分人群）；要么等一次地基级的大迁移——平台整个换代，所有人被迫重新选一次，$n^2$ 被清零重来。第一章讲的软件"一次开发无限复制"让垄断成为可能，网络效应则给这个垄断上了第二道锁。

## 社交与平台：把这条定律推到极致

Windows 还算温和的——软件装在你自己电脑上，理论上你换个系统重装一遍也能活。到了社交和互联网平台这一代，网络效应被推到了裸露、赤裸、无处可逃的程度。

为什么？因为社交产品**除了网络效应几乎什么都不是**。一个聊天软件的代码，一个像样的团队几个月就能写出功能一样的。它唯一的、也是全部的价值，就是"你想聊的那些人在不在上面"。产品可以被抄，你朋友的名单抄不走。这就是为什么无数个"体验更好的社交 App"发布即死——它们不是败给了功能，是败给了"我朋友不在这儿"。

这里还叠了一层更狠的东西。社交平台里，网络效应和上一章的转换成本合体了：你在一个平台上攒了多年的好友、群、聊天记录、发的内容——**这些不是产品功能，是你的人生存档，搬不走**。于是"别人都在这儿"（网络效应）和"我东西都在这儿"（锁定）两股力拧成一股绳，护城河深得让后来者连试都不想试。

所以你会看到这一代平台的胜负极端得可怕：同类产品里往往就一个活得很好，第二名不是小而美，是直接不存在或苟延残喘。这不是运气，是 $n^2$ 曲线的必然结果——它天生不允许并列第一。

## 但这条曲线，也能反过来碾你

本书的书名叫《护城河为什么会一夜蒸发》，网络效应恰恰是"一夜蒸发"最经典的现场，因为**让它涨的机制，和让它崩的机制，是同一个**。

$n^2$ 是对称的。人往里进，价值成对地涨；人往外走，价值也成对地塌。而且社交网络的人是有黏连的：你走不走，看你在乎的人走没走。于是可能出现雪崩——一小撮核心的人先离开，剩下的人发现"我想找的人不在了"，跟着走，再触发下一拨。曲线可以陡峭地冲上去，也能同样陡峭地砸下来。历史上不止一个曾经统治级的社交网络，就是这么在几年之内从"所有人都在上面"变成"没人记得它"，快得像塌方。

这就是网络效应最让人不安的地方：它给你的护城河，深到荒谬；可这护城河里灌的不是水，是**别人的选择**。人还在，河就在；人心一动，河一夜见底。你并不真正拥有它，你只是暂时被它选中。

那么问题来了：如果一家在位者又有网络效应、又有转换成本，看起来铁桶一块——它还能怎么被打死？下一章我们看一招最阴的阳谋：不跟你比产品，而是把你赖以收费的那个东西，**免费送给全世界**。当你的核心产品别人白给，你这门生意，连同护城河，一起归零。

# 第五章 · 把地基送人的阳谋

上一章我们看到，网络效应能让一家公司赢家通吃——用的人越多，产品越值钱，后来者几乎没法翻盘。你可能会觉得，占住了这种位置的公司就该高枕无忧了。这一章我要讲一件更狠的事：有时候杀死你的，根本不是一个更好的对手，而是一个把你的产品**白送**的巨头。你卖 100 块的东西，它捆在自己的系统里免费发。这一招没有"打不打得过"的问题——它压根不跟你比好坏，它比的是"要不要钱"。

先把这章要回答的问题摆出来：**为什么"免费+捆绑"能在两三年内绞死一家如日中天、市值几十亿美元的公司？**它靠的到底是什么力量？我们拿 1995 到 1998 年那场著名的浏览器战争当解剖标本——微软怎么用一个不要钱的 IE，把当时统治浏览器市场的网景（Netscape）从 90% 的份额打到几乎归零。

## 先搞懂一个词：什么叫"补充品"

要理解这场屠杀，你得先懂一个经济学里的概念。

补充品（complement），就是"跟你搭着一起用才有意义"的东西。热狗和面包是补充品，打印机和墨盒是补充品，汽车和汽油是补充品。它们的关系有个反直觉的特点：**当一样东西变便宜，另一样东西的需求就会涨。**汽油便宜了，大家就更爱开车、更爱买车。反过来也成立——如果有人把面包白送，热狗就能卖得更多、卖得更贵。

> 大白话
>
> 说人话：你卖的东西，如果有个"最佳搭档"，那么让搭档降价甚至免费，其实是在给你自己拉生意。所以聪明的公司有一条隐藏策略——**想尽办法把自己产品的补充品变便宜**。

现在关键的一步来了：**一个东西是谁的补充品，决定了它的命运。**同样一个浏览器，在网景眼里是"要卖钱的主产品"，在微软眼里却只是"Windows 的补充品"。这个视角差异，就是整场战争的胜负手。

## 网景做错了什么？它把补充品当主产品卖

1994 年，网景推出了 Navigator 浏览器，好用、快、漂亮，几乎人人都在用它上网。1995 年 8 月它上市，第一天股价就翻倍，成了那个年代互联网狂热的象征。这时候网景的商业模式是什么？**卖浏览器**——个人用可以先免费试用，但企业、正版用户是要付钱买 license（授权费）的，一份几十美元。

这套模式在没有对手时很赚。问题是，浏览器本质上是"上网的一层壳"，它没有第三章讲的那种转换成本（换掉它要付出的代价——数据搬不走、习惯改不了、生态离不开）。你今天用 Navigator，明天换一个别的浏览器，你的网页照样能打开，你什么都没损失。**一个换起来毫无痛苦的产品，是收不住用户的。**网景站在一片没有护城河的开阔地上，还把身家性命全压在"这个开阔地上的东西能一直卖钱"这个假设上。

## 微软的算盘：浏览器归零，我的地基就更值钱

比尔·盖茨在 1995 年 5 月写过一份内部备忘录，叫《互联网浪潮》（The Internet Tidal Wave），核心意思是：互联网会重塑一切，微软必须全力扑上去。为什么微软这么紧张一个"只是浏览器"的小公司？

因为网景的创始人放过一句让微软脊背发凉的话——大意是，将来浏览器会变成新的操作系统，Windows 会退化成"一堆没人关心的、蹩脚的驱动程序"。翻译一下这句话的杀伤力：如果所有软件都搬进浏览器里跑，那用户装的是 Windows 还是别的系统就无所谓了。**微软最值钱的地基——Windows 的垄断——会被浏览器架空。**这正是第二章讲的"平台被换"的剧本，只不过这次微软是那个怕被换掉的人。

于是微软的策略非常清晰，而且冷酷得漂亮：

- **把浏览器彻底免费。**IE 不要钱，任何人随便下载、随便用，企业也不用买 license。
- **把 IE 捆进 Windows。**1997 年起，IE 被越绑越死，最后干脆变成 Windows 的一部分——你装了 Windows，就等于装了 IE，卸都卸不干净。全世界数以亿计的电脑，开机就自带一个浏览器。

这两步合起来，对网景就是灭顶之灾。你想想：一个用起来没差多少、还免费、还预装在你新电脑里的东西，你为什么要专门去买一个网景的？**当补充品被白送，靠卖这个补充品活着**

**的公司，产品瞬间一文不值。**不是因为它做得差，是因为"要钱"这件事本身，在一个"免费且预装"的对手面前，成了致命缺陷。

> 大白话
>
> 核心机制一句话：微软和网景卖的是同一个东西，但这个东西对两家的意义完全不同。对网景，它是**唯一的饭碗**；对微软，它只是**保护主饭碗（Windows）的一次性弹药**。一方在拿命换钱，另一方在拿钱换命——赌注根本不对等，网景从一开始就输在这个不对称上。

## 为什么"免费"打得过"更好"？

这里有个值得细想的点：微软能这么干，前提是它有一个**已经赚钱的、垄断的主产品**去供养这场补贴。IE 是净亏钱的——研发、推广全是成本，收入是零。微软能承受，是因为 Windows 每年在源源不断地进账。这就像一家有金矿的公司，可以在隔壁开一家永远亏本的免费面包店，只为了让来买面包的人顺路多买它的黄金。

而网景没有金矿。它**只有那家面包店**。当巨头把面包免费发的时候，网景连跟的资格都没有——它一免费，自己就饿死了；不免费，用户就跑光了。这是一个无解的死局。回看第一章说的那条"反常物理"：软件边际成本近乎为零，复制一份不花钱。正因为如此，把软件白送几乎不增加成本——**"免费"这把刀，只有在软件世界里才这么锋利**，在实体世界你送一百万个真面包是要真金白银的。

结局大家知道了。到 1998 年前后，IE 的份额一路反超，网景节节败退，同年被 AOL 收购，浏览器业务作为一门独立生意基本终结。也是在 1998 年，美国司法部对微软发起了著名的反垄断诉讼，捆绑 IE 正是核心指控之一。**连法律都认为，"把地基送人"这招太致命，需要用国家力量来拦。**——注意，是拦"垄断者用免费捆绑"，不是拦"免费"本身。这个区别很重要：免费不违法，用一个垄断的主产品去补贴、挤死另一个市场里靠卖钱活着的对手，才是问题。

## 这套机制，今天还在反复上演

别以为这是上世纪的老故事。这个"用免费补充品绞杀单品公司"的套路，是软件史上最稳定复现的杀招之一。谷歌把地图、邮箱、办公套件几乎全部免费——因为它们都是广告业务的补充品，用的人越多，谷歌的广告越值钱，而那些曾经靠卖地图数据、卖邮箱服务收费的公

司就没了活路。每一次，剧本都一样：**某个巨头有个赚钱的主业，于是它可以把你的整个行业当成"补充品"白送出去，用你收不了的成本，换它收得回的钱。**

所以，如果你正在做一个软件产品，有一个问题值得你半夜惊醒时问自己：**我卖钱的这个东西，会不会正好是某个巨头主业的补充品？**如果是，那么它随时可能被那个巨头免费送出去——不是因为你做得不好，而是因为送你这块，能让它自己那块更值钱。这跟你的努力无关，跟你处在谁的食物链上有关。

网景的悲剧，说到底是**护城河的方向搞反了**：它以为守住"产品够好"就安全，却不知道真正的安全是守住"别人离不开你、换你要付出巨大代价"。微软 Windows 有这个（换系统的代价高到吓人），网景的浏览器没有。下一章我们要谈的，正是这条更深的护城河——当越来越多的人用同一个东西，会长出一种什么样的力量，让后来者哪怕做得更好、哪怕免费，也依然攻不进来。

# 第六章 · 免费怎么还能赚钱

先问你一个傻问题：一样东西，你不收用户一分钱，你怎么赚钱？

前五章里，软件公司赚钱的路子其实万变不离其宗——**东西卖给用它的人**。IBM 拆开卖软件、微软卖 Windows 授权、Oracle 卖数据库、Office 卖盒装光盘，都是一手交钱一手交货。哪怕上一章讲的微软免费送 IE，那也不是真的"免费"——它免费只是为了保住 Windows 这门收钱的生意，IE 本身从头到尾没打算靠自己挣钱，它是一颗子弹，不是一门生意。

但这一章的主角不一样。它把产品**永久地、彻底地、对所有人免费**，而且越多人白嫖它越高兴——然后它成了人类历史上最赚钱的软件生意之一。这不是促销，不是补贴烧钱等回本，是一种全新的赚钱结构。搞懂它，你就搞懂了为什么"商业模式的地心引力"在 2000 年前后第一次调转了方向。

## 钱是从哪个方向掉下来的

我们说的商业模式，说白了就是一句话：**谁给你钱，为了什么给你钱**。地心引力这个比方是想说——一家公司所有的动作，最后都会朝着"钱掉下来的那个方向"倾斜，就像水一定往低处流。你收谁的钱，你就会讨好谁、优化谁的体验、防着得罪谁。

前面所有公司，钱掉下来的方向都是同一个：**用户的钱包**。用户即客户，用的人就是付钱的人，这两个词在你脑子里几乎是一回事。所以做产品的人天然会想：怎么让用户觉得值这个价、怎么让他续费、怎么用文件格式和转换成本把他拴住（第三章那套）。

Google 干的事，是把这根引力线剪断，重新接到了另一个地方——**广告主的钱包**。于是用它的人和付钱的人，第一次变成了两拨完全不同的人。

大白话

这里有个反直觉的点你得先接受：一旦"用的人"不等于"付钱的人"，产品免费就不再是牺牲，而是变成了武器。因为免费的东西传播起来没有任何摩擦——不用你掏钱、不用你决策、不用你说服老板报销，你只是随手一用。用的人越多，你手里能卖给广告主的东西就越多。产品成了一台"把人变成注意力"的机器，人越多，机器功率越大。

## 但"注意力"这三个字，值钱得非常不平均

广告不是新东西。报纸、电视、路边广告牌卖了上百年，卖的都是"注意力"——你看报纸的时候顺眼扫到那半版广告，报社把你这一眼卖给了商家。互联网早期的门户网站（新浪、雅虎那种）也是这么干的，页面顶上挂个横幅图片，谁来都看得见。

问题是，这种广告有个致命的毛病：**它不知道你想要什么，只能瞎打**。电视上给全国人民播一条汽车广告，可这一刻真想买车的可能一万个人里才一个，剩下 9999 个眼神从广告上滑过去，钱就烧掉了。行内有句老话——"我知道我一半的广告费浪费了，但我不知道是哪一半。"横幅广告就是这德性，效果差，所以单价压得很低。

Google 的搜索广告，恰恰把这个百年难题给解了。关键不在"广告"，在**搜索**这个动作本身。

### 搜索是全世界最诚实的一句话

你想想，一个人在搜索框里敲下"上海 二手 卡罗拉 报价"的那一瞬间，他等于亲口、主动、清清楚楚地告诉了机器：我现在就想在上海买一辆二手卡罗拉。没有任何一种媒介能拿到这么精准的信号——电视不知道你在想啥，报纸不知道，连你老婆可能都不知道。

所以搜索广告不是"把广告塞到你面前赌你感兴趣"，而是**你先说出了需求，广告只是顺着你的需求递上答案**。这时候那家二手车行的广告，对你不是打扰，几乎是帮忙。一个想买车的人看到卖车的广告，点进去成交的概率，比电视观众高出不知道多少倍。**同样一次曝光，值的钱天差地别。**

大白话

一句话总结这个魔法：传统广告是"我猜你可能想要"，搜索广告是"你说了你想要"。前者在黑暗里开枪，后者是你自己走到枪口前举手。广告效果好几个数量级地提升，广告主自然愿意出高得多的价——而这一切的前提，是先有海量的人心甘情愿地免费来搜东西。

## Google 还干了两件让它坐稳的事

光有"搜索信号值钱"还不够，Google 把这门生意做成了一台自我加固的机器，靠的是下面这两个设计。

第一，竞价排名 + 按点击付费——广告位不是标个死价卖给你，而是让想买同一个关键词的广告主之间互相出价竞拍，而且只有用户真的点了广告、你才付钱。这两条组合起来非常狠：一方面，谁最舍得为"卡罗拉"这个词付钱，这个价就由市场自己顶到最高，Google 不用费心定价；另一方面，广告主只在真有人点、真有可能成交的时候才掏钱，风险小到他不好意思不投。这等于 Google 给成千上万广告主发了一台"稳赚不亏"的印钞机，代价是每次印钞它抽一道水。

第二，也是更本质的——它把第四章那个网络效应（用的人越多、这东西对每个人越有用）搬进了搜索里，只不过换了个更隐蔽的形式，叫**数据反馈的飞轮**：

- 越多人来搜 → Google 就看到越多"人们搜什么、点了哪条结果、又退回来重搜"的记录；
- 这些记录反过来告诉它哪个答案是好答案 → 搜索结果就越准；
- 结果越准 → 更多人来用，而且离不开；
- 更多人用 → 更多人的"需求信号"待价而沽 → 广告卖得越多、越贵；
- 广告越赚钱 → 越有钱砸进服务器和工程师，把搜索做得更好 → 回到第一条。

这个圈一旦转起来就很难被外人打破。后来者哪怕做出一个技术上不差的搜索引擎，它没有那几十亿次真实搜索喂出来的数据，结果就是没那么准；不够准就没人用；没人用就更拿不到数据。这就是为什么搜索这门生意几乎是赢家通吃——护城河不在那份搜索的代码里（代码别人也能写），而在**亿万人每天用脚投票沉淀下来的数据，和那条把注意力换成钞票的广告管道里**。

## 为什么说这是"地心引力第一次转向"

把镜头拉远，你会发现这件事的分量远不止"Google 很赚钱"。它改写了一整套关于"软件靠什么活着"的默认答案。

在此之前，一个软件人做产品，脑子里绷着的弦是：**怎么让用户愿意为它掏钱**。功能要对得起价格，不能太大方，不然没人买正版。而 Google 这套模式一旦被验证成立，一大批公司的弦就换了——**先不惜代价把用户量做到最大，钱的问题以后用广告解决**。产品免费到底、拼命降低使用门槛、甚至倒贴钱抢用户，因为在新引力下，用户不是你要变现的对象，用户本身就是你要囤积的**货**，真正的买家在另一头。

你现在熟悉的太多东西都是这根引力线接过去之后长出来的：免费的邮箱、免费的地图、免费的手机操作系统（Android 白送给手机厂商，图的也是把搜索和广告铺到每一部手机上）。它们看着是慷慨，骨子里都是同一句话——**把用户攥在手里，向他们身后的广告主收税**。

当然，这根引力线转向也留下了它的账单，只是不写在你的信用卡上：你付的"钱"是你的注意力，和你每一次搜索、每一次点击暴露出来的自己。这笔账后来引出的隐私、垄断、"用户到底是客户还是商品"的争论，一直吵到今天。但那是另一个故事了。

到这里，我们已经见过软件赚钱的两种大逻辑：**卖给用户**，和**把用户卖给广告主**。它们有个共同的隐含前提——软件本身是"我的私产"，是我关起门造出来、别人拿不走的东西。护城河，要么在于我把它卖得贵，要么在于我把用的人攥得牢。

可就在同一个年代，另一股力量正从完全相反的方向掀桌子：如果软件根本不是谁的私产，而是一群素不相识的人把源代码摊在阳光下、免费送给全世界，连你的竞争对手都能白拿去用——这样的东西，怎么可能干得过关起门来收钱的巨头？下一章，我们去看敌人是怎么给你送来一把免费的枪的。

# 第七章 · 敌人送你免费的枪

先问一个怪问题：如果我要打垮一家卖软件的公司，最狠的招是什么？降价？挖人？做个更好的产品？都不是。最狠的一招是——**把一个和它功能一样、还完全免费、连源代码都白送的东西，摆在所有人面前**。它卖 5 万美元一套，你标价 0 元，而且永远不涨价。它请不起打价格战，因为对手根本不靠这个赚钱。

这听起来像天方夜谭：谁会花几年时间写软件，然后一分钱不收地送出去，还把最值钱的源代码公开？但这件事真真切切发生了，而且它没打垮一两家公司，它掏空了一整个行业的地基——卖"服务器软件"的那门生意。这一章我们就来搞懂这股力量：**开源，到底是怎么把"把软件当私产来卖"这套祖传商业模式，连根拔起的。**

## 先说清楚，"源代码"为什么值钱

上一章我们看到，Google 让商业模式的地心引力第一次转向——产品免费，向广告收费。这一章的转向更狠，它动的不是"向谁收费"，而是"你到底卖的是不是私产"。

要理解开源的杀伤力，得先理解它砸的是什么。软件公司卖你一个程序，给你的是编译好的二进制文件——就是那堆机器能跑、但人看不懂的 0 和 1。真正的原材料，源代码（程序员用人类能读的语言写的那份"配方"），公司锁在自己保险柜里。你买了 Windows，你有房子住，但你没有图纸，不能改一砖一瓦，更不能自己再盖一栋送邻居。

> 大白话
>
> 说人话：传统软件公司卖的不是"软件本身"，是"只有我有配方"这件事。第一章讲过，软件复制的边际成本几乎为零——多卖一份不多花钱。那凭什么还能收 5 万？凭的就是"配方在我手里，你只能来买"。护城河的名字，叫**代码所有权**。

开源做的事，就是把这条护城河直接填平：开源软件（open source，源代码公开、允许任何人免费使用、修改、再分发的软件）不但白给你二进制，还把配方连同"你随便改、随便传"的许可一起塞给你。当配方不再稀缺，"只有我有配方"这个护城河，一夜蒸发。

## 免费的枪，是敌人自己送上门的

本章标题不是修辞。开源的枪，很多时候是被它打垮的那些公司，自己一手递出去的。

看服务器这块地。上世纪 90 年代末，网站要跑在服务器上，服务器要有操作系统，还要有一个专门"接客"的程序——Web 服务器（负责接收浏览器请求、把网页发回去的软件）。这两样东西，本来都是能卖大钱的商品。Sun、微软这些公司卖操作系统授权，一台服务器几千上万美元。

然后两个免费的东西来了：

- Linux——一个由一位芬兰大学生 1991 年起头、随后全世界成千上万程序员一起接力写出来的免费操作系统内核。谁都能下载，谁都能装在任意多台机器上，一分钱不花。
- Apache——一个免费的 Web 服务器程序。它出现后没几年，就成了全世界过半网站背后跑的那个程序，长期是市场占有率第一。这个"过半"，是响当当的事实。

结果是什么？一门"卖操作系统给服务器"和"卖 Web 服务器程序"的生意，被两个价格永远是 0 的东西，从下面掏空了。你没法跟 0 竞争。你降到 100 美元，对手还是 0；你送免费试用，对手连"试用期"这个概念都不需要。

更要命的是那个"敌人送枪"的机制。Apache 最早的一批代码，来自一个叫 NCSA 的机构做的、公开出来的 Web 服务器程序；一群站长因为原项目没人维护了，各自打补丁("a patchy server"这个名字的段子就是这么来的)，干脆凑一起接着搞。也就是说，**这些免费武器往往是从别人放弃、或主动公开的成果上，一点一点长出来的**。你今天开放的一段代码、你培养出来的一批懂行的程序员、你为了拉拢开发者而公开的接口，都可能变成明天砸向你自己的那把枪。

## 为什么"一群人免费写"能干过"一家公司花钱写"

这里有个反直觉的点值得停一下：凭什么一盘散沙、没人发工资的志愿者，能写出干翻商业巨头的软件？

关键在软件这门生意的一个独特性质——第一章说过，它复制近乎免费，同时它还有一个孪生性质：**修改是可以叠加的**。我修好一个 bug，把这份改动传出去，全世界立刻都受益，谁都不用再修第二遍。传统公司里，一个 bug 只有你们几十个员工能碰；开源项目里，理论

上全世界会用它的人都能碰。有句在程序员圈里流传很广的话，大意是：**只要看的人足够多，再深的 bug 也会变浅**——因为总有一双眼睛正好踩过这个坑、正好知道怎么填。

> 大白话
>
> 说人话：闭源公司是"一队人对全世界的需求"，开源项目是"全世界对全世界的需求"。当一个软件用的人越多，愿意顺手贡献一点改进的人也越多——这几乎是把第四章的**网络效应**搬到了"生产"这一端。用的人多 → 改的人多 → 软件更好 → 用的人更多。滚起来之后，一家公司的研发预算再高，也追不上一个全球社区的下班时间总和。

当然，不是所有软件都适合这么干——越是"基础设施"、越是大家需求高度一致的东西（操作系统、Web 服务器、数据库、编程语言），越容易被开源攻陷，因为一万个人想要的东西高度重合，合力才使得上。越是花哨、口味各异的终端产品，开源反而不占优。这条边界，后面会有用。

## 微软：从"开源是癌症"到亲手拥抱

没有谁比微软的态度转弯，更能说明这股力量有多硬。

2001 年，微软当时的 CEO 史蒂夫·鲍尔默公开放话，把 Linux 说成是"附着在它碰到的一切知识产权上的**癌症**"。这不是一时口快。微软怕的正是本章讲的机制：它的整个身家，就建立在"代码是我的私产、你只能付费授权"这套逻辑上（第二、三章讲的平台和锁定，都得先立在这个地基上）。开源恰恰否定了这个地基本身。对当年的微软来说，开源不是一个竞争对手，是一种否定它存在方式的"病"。

可是二十年后呢？微软自己成了全世界在开源社区里贡献最多的公司之一；它花 75 亿美元买下了全球最大的开源代码托管平台 GitHub——就是那个全世界程序员放开源代码、协作改代码的地方；它甚至让 Linux 能直接跑在 Windows 里。当年骂"癌症"的公司，如今把开源捧成了核心战略。

这不是良心发现，是算清了账。这就引出本章最要紧的那句话：

> 开源没有消灭护城河，它只是**把护城河从"我独占代码"这个位置，赶到了别的地方去**。

## 护城河没消失，它搬家了

当代码本身不再值钱、谁都能白拿，钱和权力就流向那些开源填不平的洼地。微软和后来所有活得好的公司，都是想明白了这几个新落脚点：

1. **运维和托管。**软件白送，但把它稳稳当当地跑起来、不宕机、能扩容、出事有人半夜爬起来修——这件事又难又要命，值钱。你可以免费下载数据库，但你多半宁愿花钱租一个"别人替你操心"的版本。这正是第十章要展开的云生意的种子。
2. **信任、合规与"有人担责"。**一家银行敢不敢把核心系统压在"网上一群志愿者写的免费代码"上？它需要一个能签合同、出了事能索赔、承诺长期支持的对象。有公司专门靠"给免费的 Linux 提供付费保障"活成了大生意——卖的不是代码，是**一个能负责任的名字**。
3. **接口与数据。**代码开源，但真正卡住你的往往不是代码，是你的数据在谁手里、你习惯了谁的接口——这又回到了第三章的锁定。

所以微软的转身其实很冷静：既然守不住"独占代码"这条河，那就顺水推舟——用开源去拉拢全世界的程序员、把大家都吸到自己的云和平台上，再在"托管、服务、信任"这些新洼地里收费。它没有战胜开源，它学会了**站在开源的上游收税**。

## 这一章到底说了什么

把一件事讲透，无非这么几句：软件公司祖传的护城河，是"配方只有我有"。开源用"配方白送、还允许你随便改"这一招，把这条护城河填平了，还借助"用的人越多、帮忙改的人越多"这股类似网络效应的力量，让一群不拿工资的人干翻了拿工资的公司——尤其在操作系统、Web 服务器这类大家需求高度一致的地基软件上。护城河没有消失，它搬去了运维、信任、接口和数据这些开源填不平的地方。看懂了这一点，你就明白微软为什么能从骂"癌症"到砸 75 亿买 GitHub：不是它变善良了，是它把河改道了。

但请注意，本章有个前提一直没被挑战：我们默认"更好、更省、被更多人认可的东西，最后会赢"。开源赢，是因为它对用户更划算。可现实里有一类怪事恰恰相反——**一个又烂又便宜、大客户看不上的新玩意儿，反而能把如日中天、什么都做对了的巨头掀翻**。而且巨头不是蠢，是它越理性、越会算账，就越必然错过它。下一章，我们就去拆这个让无数聪明公司体面地走向死亡的陷阱。

# 第八章 · 大公司为什么打不过小公司

上一章我们看到，开源把"代码是我的私产、你得花钱买"这件事连根拔起——护城河从"谁拥有代码"漂走了。但那至少还是一场看得见的战争：微软知道 Linux 是敌人，只是打不赢。这一章要讲一件更别扭的事：**敌人根本没露脸，你就已经输了；而且你每一步都做得很对，正是这些"对"把你送进坟墓。**

先抛一个反直觉的问题。一家公司要是又蠢又懒，被小公司干掉，天经地义，没什么好研究的。可现实里最经典的败局是反过来的：那些管理最精良、最听客户话、利润最高、被写进商学院教材当正面教材的公司，恰恰是被一个一开始又烂又便宜、根本上不了台面的小东西掀翻的。为什么"优等生"反而系统性地栽在同一个坑里？这个坑有名字，叫创新者的窘境——一家好公司之所以失败，不是因为它做错了，而是*因为它做对了*。

## 先看那条要命的曲线

要讲清这件事，得先认识一种东西叫颠覆式创新（disruptive innovation）。注意，它说的不是"更厉害的技术"。恰恰相反——它一登场，各项指标往往比现有产品*差*。

你可以把它想成两条往上爬的斜线。横轴是时间，纵轴是"性能"（比如硬盘容量、相机像素、CPU 速度）。

- 第一条线是**技术进步的速度**：工程师每年能把产品做得多好。这条线通常很陡——技术进步比你想象得快。
- 第二条线是**客户真正用得上的速度**：一个普通客户每年多消化多少性能。这条线通常很平——一个人一年能多用的性能是有限的。

关键在于：**第一条线比第二条陡。**意思是，早晚有一天，产品好得*超过了客户的需要*。这时候奇怪的事就发生了：性能不再是卖点了。你的硬盘从 1TB 做到 2TB，客户耸耸肩——他 500GB 都没用满。你辛辛苦苦攒的技术优势，客户不肯为它掏钱了。

而颠覆者，就从下面那条线的*下方*钻进来。它一开始很烂，连最低要求都够不着，所以只能待在没人要的犄角旮旯里。但它便宜、简单、有别的好处（更小、更省电、更好玩）。只要它自己也在往上爬——而且它爬得比"客户需要的速度"快——那么总有一天，它会从下面追

上那条"够用线"。一旦它够用了，游戏就结束了：客户不再需要你那身多余的本事，转身选了那个更便宜更省心的。

> 大白话
>
> 翻译成人话：你被干掉，不是因为对手比你强，而是因为你早就"强过头"了。你在客户已经不在乎的维度上一路狂奔，把身家全押在那儿。对手在一个你看不上的、烂到不值得竞争的维度上悄悄追赶，等你回头，够用线已经被它踩在脚下了。

## 为什么是"理性"把你害死

这里是全章最要命的一点，也是它区别于"大公司变笨了"这种廉价解释的地方。在位者错过颠覆，**不是因为看不见，而是因为看见了也躲不开**——躲开它反而是不理性的。

假设你是那家大公司的产品经理。有人拿来一个新玩意儿，说这是未来。你拿数据一算：

- **你最大的客户不要它。**他们要的是更快更强，这新东西又慢又弱，拿给他们等于侮辱。听客户的话——这可是所有管理书教你的第一条——就得拒绝它。
- **它利润薄得可怜。**你现在的产品毛利也许有 60%，这新市场小、单价低、毛利也许只有 20%。把工程师和产能从 60% 的生意挪到 20% 的生意上，你的财务总监会当场把提案摔在你脸上。这在数学上就是错的。
- **它市场太小，喂不饱你。**一家几百亿营收的公司，需要的是能贡献几十亿的新增长。一个刚冒头、总共才几千万的市场，就算你 100% 吃下也不够塞牙缝。理性的资源分配，一定把钱投向大机会，也就是继续伺候现有大客户。

你看，**每一条都对**。听客户的、追高毛利、把资源投向大市场——这正是让你成为行业老大的那套本事。可就是这套本事，让你在每个决策路口都*正确地*、*理性地*选择远离那个未来。等到那个小东西长大、够用线被它踩穿，你想掉头，晚了：技术积累、渠道、成本结构、组织习惯全站在旧世界那边。这就是"窘境"二字的分量——不是选错了，是**没有一个对的选项**。你的优点就是你的死因。

## 软件业把这个坑挖得更深

这套机制最早是哈佛的克里斯坦森从*硬盘*行业里挖出来的（1990 年代，他发现每次硬盘尺寸从 8 英寸缩到 5.25 再缩到 3.5、2.5 英寸，在位霸主几乎无一幸免，全被新尺寸的新厂商取代——尽管每一代小硬盘刚出来时容量都远不如大的）。但真正把这个坑挖到深不见底的，是软件。

回想第一章我们讲过的软件的怪脾气：**边际成本几乎为零，一次开发无限复制。**这个特性给颠覆者装上了火箭。硬盘颠覆者还得建工厂、造实体、铺物流，追赶那条"够用线"要花好几年、烧掉真金白银的产能。软件颠覆者呢？它追赶得几乎不花钱——写一次，全世界随便下载。它可以先免费送出去（第五、六章的把戏），靠着零复制成本把用户攒到几百万，再慢慢补上功能。在位者眼里那个"又烂又免费、根本不赚钱"的东西，正因为不赚钱、正因为它踩着软件的经济学，才最凶。

最教科书的一幕发生在移动端。一整代桌面软件霸主，做梦都想不到自己会输给手机——那块小小的、又慢又卡、屏幕只有巴掌大、连个像样键盘都没有的破玩意儿。用当年任何一个理性 PC 产品经理的标准看，手机上的软件全是*降级*：功能少、算力弱、屏幕小。它稳稳落在"够用线"下方。于是巨头们理所当然地把它当成 PC 的一个可怜附属品来对待——桌面才是正经生意，利润和大客户都在那儿。

可手机沿着它自己的斜线往上爬，而且爬得飞快。它带来了桌面永远给不了的东西：随身、随时在线、有摄像头、知道你在哪。几年之内，那条线追上并踩穿了"够用线"——对绝大多数普通人来说，一部手机上的软件**已经够用，甚至更好用**。等桌面巨头反应过来要转身，用户的注意力、开发者的热情、新的平台规则（还记得第二章的地基吗？）已经统统搬到了一块他们既没地基、也没手感的新大陆上。他们没做错任何一步，只是每一步都把自己更牢地钉在了旧大陆。

大白话

所以别再问"这么大的公司怎么会那么傻"。它们不傻，正好相反，是太聪明、太守规矩了。它们精确地计算了投入产出，精确地服务了最赚钱的客户，然后精确地把自己算进了坟墓。真正的教训冷得刺骨：**让你赢下现在的每一条准则，都在悄悄地让你输掉未来。**

## 那到底怎么破

克里斯坦森给的解法本身也反直觉，而且很少有人真做得到：既然主流组织的每一根神经都在*正确地*排斥颠覆，那就**别在主流组织里搞它**。单独拉一支队伍，给它独立的预算、独立的损益表、甚至允许它去革自己母公司的命。让它去追那个小到看不上的市场，用那份薄到看不上的利润活着，别让它跟核心业务抢资源、比毛利——因为一比,它必输,一输就会被理性地砍掉。

这很难。难不在技术，在人性和账本：你得眼睁睁看着一个亲儿子部门，去啃利润最薄的骨头，去做一个可能干掉你现金牛的东西，还得护着它别被公司里所有"理性的人"掐死。能扛住这份别扭的公司，屈指可数。

到这里，我们已经攒齐了好几股力量：零边际成本（一章）、平台地基（二章）、锁定（三章）、网络效应（四章）、免费绞杀（五章）、广告与开源改写的赚钱方式（六、七章），再加上这一章的窘境。它们大多在讲"护城河怎么被冲垮"。但下一章，SaaS（把软件从"买断一次"改成"按月租"的商业模式）会告诉你一个新玩法：与其提心吊胆守着一条随时会蒸发的护城河，不如**换一种收钱的方式，让客户每个月都得回来续命**——地心引力,又要转向了。

# 第九章 · 从'买断'到'月租'

上一章我们看到，在位的大公司不是因为蠢才输，恰恰是因为它太聪明、太尽责地服务着现有的大客户，才必然错过那个"又烂又便宜"的新东西。这一章要问的问题更别扭：**就算你的产品没被颠覆，行业里也可能有人不改一行核心代码，就把你原来赚钱的方式给废掉。**

废掉的不是代码，是"怎么收钱"这件事。

## 先看一个反直觉的现象

1999 年之前，你买软件的样子是这样的：花一大笔钱，拿到一张光盘（或者一摞），装到自己公司的机器上，这软件就永远是你的了。这叫 买断制授权（perpetual license）——付一次钱，永久使用权归你。听起来天经地义，对吧？就像买冰箱，买完就是你的。

然后有家公司站出来说：这套我们不卖了。软件不装在你机器上，装在我这儿，你也不买断，你按月租。奇怪的是——它不但没饿死，还逼得整个行业几年之内全部跟着改成了租的。今天你用的几乎每一个软件，从公司里的 CRM 到你手机上的会员，都是月租或年租。

问题就在这儿：**如果买断对客户明明更"划算"（付一次就完事），为什么整个行业会集体倒向那个让你永远付钱的模式？**

## 买断制的账，其实两边都难受

先别急着同情客户。买断制这笔买卖，卖方和买方其实都在硬扛一些说不出口的痛。

对**客户**来说，那"一大笔钱"是要一次性掏出来的。一套企业软件动辄几十万上百万美元，这不是买软件，这是上一个大工程：

- 软件光盘只是开始，你还得买服务器、建机房、雇一队 IT 的人去装、去调、去打补丁。真正花的钱，软件本身可能只占一小块。
- 装完不代表能用。企业软件的部署常常拖上一年半载，上线那天发现和业务对不上，钱已经花了，退不了。
- 你买的是"这一版"。明年出了新版本？对不起，那是另一笔升级费。

对**卖方**来说，账更刺激。买断制的收入长这样：签下一个大单，一次性确认一大笔钱，然后……然后就没了。这个客户明年可能一分钱不再给你。于是每年年初，你的销售额都要**从零开始**。去年做到一个亿，今年一月一号，账面归零，重新卖。

> 大白话
>
> 这就像开饭馆，但规矩是每桌客人只能来一次、付一次账，从此再不能进门。你去年生意再好，今年还得满城去拉新客人，一个都不能重复。你永远在追新客户，追不到就当场饿死——哪怕你的菜没变差。

## SaaS 干的事：把"一次性"拆成"细水长流"

Salesforce 1999 年在旧金山成立，创始人 Marc Benioff 喊出的口号是两个字加一个禁止符号：No Software——"不要软件"。这话听着像行为艺术，一家卖软件的公司说"不要软件"。他的意思不是不给你软件用，而是：不给你光盘、不用你装、不占你机房。你打开浏览器，登录，就能用。

这种东西后来有了名字，叫 SaaS（Software as a Service，软件即服务）——软件不再是你买回家的一件"东西"，而是像自来水一样,你拧开龙头就用、按用量付费的一项"服务"。软件跑在供应商的机器上，所有客户共用同一套代码（这叫 多租户 multi-tenant，一栋楼里住很多户人家，共用水电管道，但各家门一关互不打扰），你只是租了其中一间。

关键的转向发生在收钱方式上。**把一笔一次性的大钱，拆成了每个月的一小笔，永远收下去。**

这一拆，看似只是把钱切成小块，实际上把整个生意的物理性质都换了。我们一条条看它换掉了什么。

### 一、客户的账一下子好算了

不用一次掏几十万，不用买服务器，不用建机房，不用等一年上线。一个月付几十块每人，明天就能用，不喜欢下个月就停。**决策的门槛从"上一个大工程"降到了"办一张会员卡"。**这意味着卖软件不再只能卖给有钱有 IT 部门的大公司，中小公司第一次也买得起、也敢买了——市场瞬间大了一圈。

## 二、卖方的收入从"归零重来"变成了"垒起来"

这是最要命的一处。订阅制下，只要客户不退，去年的收入今年**自动还在**。这叫 经常性收入（recurring revenue）——不用重新卖就会自己续上的钱。

> 大白话
>
> 还用饭馆打比方：买断制是"每桌只能来一次"；订阅制是"办月卡的老顾客，只要不退卡，每个月的钱自动到账"。于是你今年的起点不再是零，而是去年所有没退卡的人。你只要让退卡的比拉新的少，收入就是一层一层往上垒的。垒到后来，光是"老顾客不退"这一项，就够养活整个公司。

行业里于是多了一个所有人天天盯的数字：流失率（churn）——每个月有多少比例的客户退租走人。买断制时代根本没这个概念，因为客户走不走跟你今年收入没关系（钱早收了）。到了订阅制，churn 就是命门：漏水的桶，你灌得再快也存不住水。整个公司的注意力，被这个模式硬生生从"签大单"扭到了"别让人跑"。

## 三、"持续交付"从成本变成了武器

买断制下，卖方最怕的就是给你免费升级——那是纯成本，改好了也收不到新钱，不如攒着做成"新版本"再卖你一遍。所以你会看到一个别扭的现象：软件公司并不那么急着把产品做得更好，因为"更好"要留到下一次卖钱。

订阅制把这个激励整个翻了过来。因为软件跑在供应商自己的机器上，所有人共用一套代码，供应商可以**随时、悄无声息地给所有客户同时更新**，你甚至察觉不到。更关键的是：既然客户随时能退，那"让产品每个月都比上个月好一点"，就从成本变成了续命的刚需——你不持续变好，客户下个月就走。

于是这里有个漂亮的因果闭环：**因为改成了月租，卖方才有动力持续把产品做好；因为持续做好，客户才更不想走；越不想走，月租收得越稳。**模式、产品、绑定，三者互相咬合，转起来了。

## 为什么这是一种"更深的绑定"，而不只是换个收费方式

第三章我们讲过转换成本——锁住你的不是产品本身，是你换走要付的代价。SaaS 把这种绑定又加深了一层，而且加得很隐蔽。

你的数据在人家的机器上。你公司几年积累的客户资料、销售记录、工作流程，全长在这套系统里。你每个月付的那笔小钱，交换的其实是"我继续把命根子放在你这儿"。要搬走？数据导出、流程重建、全员重新培训——账一算，比每月那点租金贵得多。于是月租看着便宜，退租却越来越难。**钱是细水长流，绑定是越绑越死。**

这也解释了为什么华尔街后来给 SaaS 公司的估值高得离谱。一家买断制公司，你没法预测它明年能卖多少，因为得从零再卖一遍；一家 SaaS 公司，只要 churn 低，它明年的收入你今天就能八九不离十地算出来——**可预测的、会自己续的现金流，值钱。**市场不是在为软件付钱，是在为"这笔钱明年还在"这件事付钱。整个行业的估值标尺，被这个模式重新画了一遍。

## Salesforce 到底逼了谁

Salesforce 一开始只是做一件小事：把销售人员用的 CRM（客户关系管理）——记录"哪个客户聊到哪儿了、该谁跟进"的软件——搬到网上租着用。当时这块地盘的老大是 Siebel，卖的正是典型的买断制大套装：贵、难装、上线慢。Salesforce 就用"打开浏览器就能用、按月付"的方式，从 Siebel 服务不好的中小客户那儿一点点啃。

真正的杀伤力不在于它抢了多少客户，而在于它**证明了一件事能成**：软件可以不卖光盘、只租使用权，而且公司能活、能长大、能上市。一旦这条路被走通，所有原本靠买断制吃饭的公司都陷入了一个残酷的选择——要么守着"一次性大单"的旧账本继续舒服几年，要么自己动手，把自己那台每年归零的收银机，换成会自动续费的那台。

这几乎是第八章那个窘境的翻版，只不过换了个战场：颠覆你的不是更强的技术，是一种更黏、更稳、投资人更爱的**收钱方式**。守旧的短期利润更好看，改租的短期反而难受（收入要从"一次到账"变成"细水慢慢来"，账面上先掉一大块）。于是又一次，理性的守成通向缓慢的失血，而咬牙转型的人，几年之内把整个行业的规矩改了。到今天，"买断"几乎成了古董，"月租"成了默认。

## 把这一章收进那条主线

回到全书那个问题：决定软件公司几十年兴衰的，是哪几股反复出现的力量？这一章讲的是其中很特别的一股——**不是产品变了，是"怎么收钱"变了**。第六章我们见过一次这种转向（从向用户收钱，转向向广告收钱）；这一章是第二次（从一次性买断，转向永续订阅）。每一次收钱方式的转向，都会悄悄重估谁强谁弱：旧模式下的赢家，可能在新账本里一文不值。

但订阅制也留了个尾巴没解决。软件跑在"供应商自己的机器上"——这些机器谁来提供？机房、服务器、带宽这些最底下的地基，又是谁在掌握？Salesforce 们把软件变成了自来水，可自来水厂脚下的那片地，正在被另一家公司悄悄买下、变成所有人都绕不开的水电。下一章，我们就去看那个把"计算"本身变成水电、然后站在所有人头顶上收税的角色——**云**。

# 第十章 · 谁掌握了地基本身

上一章我们看到，Salesforce 把"买断软件"改成了"按月租"，让所有软件公司重新学做生意。但那时候，还有一件更底层的事没人注意：**租软件的公司，自己又在租什么？**

你写了一个网站，要让别人访问，得有一台开着的电脑接住请求。2005 年前后，一家创业公司想上线，流程大概是这样的：估算峰值流量，买一批服务器，找机房托管，雇人装系统、配网络、盯着别烧了。等这一切弄完，好几个月过去了，钱也花掉一大截——而这时候你还没写一行真正的业务代码。

这就是本章要回答的问题：**当"一台开着的电脑"从一件要你自己去买、去装、去养的固定资产，变成一个随开随关、用多少算多少的东西之后，谁站到了所有软件公司的下面？**他又凭什么向上面每一个人收税？

## 把计算变成水电，是什么意思

我们说一句话说滥了：云计算就是"把计算变成水电"。但"水电"这个类比到底精确在哪，很多人没想清楚。它精确的地方有三点，缺一不可：

- **按用量计费**。你家开一小时空调付一小时电费，不用为"整个发电厂"买单。云也是——你跑一台服务器一小时，付一小时的钱，关掉就不付了。
- **瞬间可得，近乎无限**。你把插头插进墙，电就来了，你从不担心"电网今晚会不会没货"。云也承诺：你想要 100 台机器，几分钟内就有；想要 10000 台，也有。
- **你完全不需要懂它怎么造出来的**。你不知道电是煤烧的还是核裂变的，也不关心。你只面对墙上那个标准插座。

第三点是最要命的。因为一旦一个东西变成了标准插座，插座后面那套巨大的、复杂的、烧钱的机器，就**和你彻底无关了**——它属于插座的主人。这就是层级上移的种子。

大白话

在没有云之前，每家软件公司都得自己"发电"：自己买机器、自己养机房。云的意思是——从此有一家公司专门发电，剩下所有人只管插插座。发电的那家，站到了所有插插座的人下面。

## AWS：一个内部工具怎么长成了万物的地基

最有意思的地方在于：AWS（亚马逊云服务，Amazon Web Services）一开始根本不是拿来卖给外人的。它是亚马逊自己被逼出来的。

2000 年代初，亚马逊是个电商公司。每上一个新功能——推荐、支付、库存——团队都要重新搭一遍底层的存储、计算、数据库，慢得要死。于是亚马逊内部做了一件事：把这些"每次都要重搭"的底层能力，做成标准化的、谁都能调用的服务。你要存东西，调存储服务；你要算力，调计算服务。不用管它背后是哪台机器。

做着做着他们意识到一件反直觉的事：**我们为自己解决的这个麻烦，全世界每一家软件公司都有**。既然我们已经把发电厂建起来了，为什么不把多余的电卖出去？

2006 年，AWS 对外开放了两样东西：S3（一个你想存多少文件就存多少、按量付费的仓库）和 EC2（你想要一台服务器，点一下，几分钟后它就在那儿运行，用完关掉）。定价简单粗暴：存了多少、跑了多久，就付多少。没有预付，没有合同，没有销售来烦你。

对当时的创业者来说，这不是"便宜了一点"，而是**那道要命的前置门槛整个消失了**。原来上线要几个月、要几十万启动资金；现在一张信用卡、一个下午。那几年冒出来的一大批公司——包括后来的 Netflix、Airbnb、Dropbox——很多都是直接长在 AWS 上的，它们从没自己买过机房。这不是巧合：**是地基变便宜了，上面才能一下子长出这么多楼。**

## "谁是平台谁定规则"，第几次重演了

如果你读到这儿觉得眼熟，那就对了。第二章里，微软靠 Windows 成了所有 PC 软件的地基，然后它定规则、抽地基、甚至自己下场做应用（第五章的 IE）。现在换了个楼层，同一出戏又演了一遍——只不过这次站在最底下的是 AWS。

平台方的权力，永远来自同一件事：**你活在它上面，而它不活在你上面。**具体到云，这份权力体现在几个地方：

### 一、它能看见谁在赚钱，然后自己下场

这是所有平台的经典动作。有个流传很广的说法叫"你的毛利就是我的机会"——意思是，你在平台上卖东西赚的那份差价，在平台方眼里就是一块它可以自己吃掉的肥肉。AWS 上跑着成千上万个数据库、缓存、消息队列产品，其中哪个火了、哪个赚钱，AWS 比谁都先看

见——因为流量就从它的机器上过。看见之后，它常常会推出一个自己的、深度集成的对应服务。你辛苦教育了市场，它坐享其成。这跟当年应用公司在 Windows 上做出好东西、然后发现微软自己做了一个内置版，是一模一样的恐惧。

## 二、它悄悄给你换了一种新的锁

还记得第三章讲的转换成本（换一家供应商要付出的迁移代价）吗？云把这把锁做得更隐蔽了。当你只是租"一台裸机"时，其实你还挺自由——机器到处都能租，搬走不难。

但 AWS 不只卖裸机。它卖几百种"省事的现成服务"：托管数据库、消息队列、身份验证、机器学习……每一个都特别好用，能让你少雇几个工程师。可你越是把业务缝进这些专有服务里，你就越搬不走——因为别家没有一模一样的东西，搬走意味着大段代码要重写。**它不是用合同锁你，是用"方便"锁你。**你是自愿走进去的，而且每一步都觉得自己占了便宜。

还有一个更直白的招数，圈内叫 egress fee（数据"出门费"）：把数据搬进 AWS 基本免费，甚至它欢迎你搬；可你要把大量数据搬出去，就得按流量掏一笔不小的钱。进门不要钱，出门收买路钱——这本身就是一道墙。

> 大白话
>
> 裸机是"随时能退的酒店"。托管服务是"装修到一半、水管电线都嵌进你家墙里的房子"。前者你说走就走，后者你搬一次家等于重装修一遍。云平台的真正护城河，不在机器，在这些嵌进你墙里的管线。

## 三、它是收税的，不是抽成的

这是我最想让你记住的一点，也是"层级上移"最狠的地方。

一个平台如果只是从某一类交易里抽成（比如应用商店抽三成），它的地盘再大也是有边的。但 AWS 收的是**另一种钱**：无论你上面跑的是电商、是游戏、是 AI、是银行、是给医院做的软件——只要你在计算、在存储、在传数据，你就在给 AWS 交钱。它不在乎你的生意是什么，它只在乎"有计算发生"。而现代几乎所有生意的底层,都是计算。

这就叫**向上收税**：它不跟你抢某一个市场，它站在所有市场的下面，从每一个市场经过的每一度"电"里抽一点。第九章那些逼你改成订阅制的 SaaS 公司,自己转身就成了 AWS 的租客

——Salesforce 向它的客户收月租，而它自己（以及无数同类）又在向云交着这笔电费。地心引力又下沉了一层。

## 那么，地基本身牢不牢？

你可能会问：既然掌握地基这么无敌，是不是意味着 AWS 会永远赢下去？

没那么简单，而这正是全书那个反复出现的张力。规模确实给了云厂商巨大的成本优势——它一次买几十万台服务器，单价你我永远拿不到，这是实打实的护城河。但也别忘了：**越标准的东西，越容易被替换。**"一台服务器一小时"这种最基础的东西，几家云厂商卖的几乎没差别，于是它们只能拼价格，利润薄得像纸——所以它们才拼命往上加那些"省事又难搬走"的托管服务，因为只有那里才有真正的锁和真正的利润。

更微妙的是，当年那些为了省心把整个身家搬上云的公司，几年之后开始算另一笔账：当我的规模足够大，租金累计起来，是不是比自己重新买机房还贵？于是出现了一小股"下云"的逆流——个别用量巨大的公司，把业务从公有云搬回自己的机房，声称省下了大笔开销。这不代表云错了，它只是提醒我们那条贯穿全书的老规律：**没有哪种护城河是永恒的，任何一层地基，都可能在成本结构变化时被上面的人重新掂量。**

更大的一记闷棍还在后头。云厂商把"计算"标准化成了水电,收了十几年太平税。可就在这几年，一种新的负载突然压了上来——训练和运行大模型需要的不再是普通 CPU，而是成堆的、极其昂贵的专用芯片。谁攥着这些芯片，谁就在云的下面又垫了一层新地基。你以为掌握了地基本身，结果地基下面，又冒出了地基。

这，就是我们下一章的故事。

# 第十一章 · 转过身的巨人

上一章我们看到，云把计算变成了水电——谁掌握了地基本身，谁就能向上面所有人收税。AWS 就是那个把水电卖给全世界的公司。现在换个问题：**如果你就是那个躺在旧地基上、日子过得极其滋润的巨头，你敢不敢亲手把自己那口井填了，去挖一口新井？**

第八章讲过创新者的窘境——大公司不是笨，恰恰是因为它太"聪明"、太听话地服从利润，才会眼睁睁错过颠覆自己的新东西。逻辑是这样的：新技术刚冒头时又小又烂又不赚钱，你手下每一个理性的经理、每一份季度财报、每一个大客户，都在喊"别碰那玩意儿，把资源投到现在这棵摇钱树上"。于是巨头一步步走进坟墓，而且每一步都走得合情合理。

这一章讲的是那个几乎不存在的反例：**一头巨兽，主动把自己最赚钱的那条腿锯掉，转身扑向那个还在赔钱的新平台，然后活了第二回。**微软就是这个反例。它凭什么能做到别人做不到的事？

## 先看这头巨兽有多难转身

2013 年前后的微软，是一台围着 Windows（那个装在几乎每台个人电脑上的操作系统）造出来的印钞机。整个公司的世界观是：世界的中心是 PC，PC 的中心是 Windows，Windows 上跑着 Office（Word、Excel 那一套），你想用微软的任何东西，先得买一台 Windows 电脑。

这套逻辑赚了二十年钱，但它有个致命的隐藏假设：**人们会一直坐在 PC 前面。**可 2010 年之后，人们的注意力搬到了手机上，而手机上跑的是苹果的 iOS 和谷歌的安卓——两个跟微软没半点关系的地基。微软在手机上几乎全军覆没。更要命的是，一旦你的软件不非得在 Windows 上跑，Windows 这个"必经之路"的价值就开始漏气。

大白话

打个比方：微软过去像一家开在唯一一座跨江大桥收费站旁边的公司，所有人过江都得从它门口走。突然有人在下游又修了两座桥（手机），还有人发明了轮渡（云）。微软的问题不是收费站不赚钱了——它还很赚钱——而是走这座桥的人开始变少，而公司里所有人都还在拼命给这座桥刷漆、加宽、装霓虹灯。

这时候摆在微软面前的选择，就是标准的窘境：继续保护 Windows（短期财报好看、所有人都舒服），还是承认"未来不属于 Windows"、把资源挪到别处（短期难看、内部所有人跟你拼命）。

## 转身的那一下，到底转的是什么

2014 年 Satya Nadella（萨提亚·纳德拉，微软第三任 CEO）上台。人们通常把微软的翻身讲成一个"新领导有远见"的英雄故事，但那是最没营养的解释。真正值得看的，是他动了哪几根**结构性的筋**——因为窘境是结构造成的，光有决心没用，你得改结构。

### 第一刀：砍掉"必须先买 Windows"这个前提

纳德拉上任没多久，就在台上演示微软把 Office 搬到了 iPad 上——苹果的平板，微软的头号对手的地盘。这在老微软是不可想象的：白送敌人平台一个杀手级应用，等于承认"你不用我的地基也行"。

但这正是关键的一步棋，它背后是一个冷酷的算账：**与其守着"只有 Windows 用户才能用 Office"这条正在缩水的护城河，不如让 Office 出现在每一块屏幕上，把它自己变成护城河。**Windows 从"你必须站上来的地基"，被降级成了微软众多产品里的一个。这就是主动自我蚕食——亲手削弱自己最值钱的那个绑定关系。

### 第二刀：把身家从"一次性卖软件"改成"按月收租"

第九章讲过 SaaS 订阅——软件不再是买断，而是像交水电费一样按月租。微软把 Office 变成了 Office 365：你不再花几百块买一个"2013 版"用到天荒地老，而是每月付费，永远用最新版。

为什么这是自我蚕食？因为它主动杀死了自己一个极赚钱的老生意——每隔几年逼你掏一大笔钱升级新版本。改成订阅，短期账面收入会难看（一大笔变成一小笔一小笔），华尔街最初也不喜欢。但它换来的是：客户装进了一条只出不进的管道，收入从"每隔几年赌一次用户会不会升级"变成"每个月都稳稳到账"。

### 第三刀，也是最重的一刀：把公司的命押到云上

真正的重注是 Azure——微软的云计算平台，对标上一章的 AWS。微软几乎把全公司的战略重心、销售队伍、研发预算，全部压到了"卖水电"这门新生意上。

这里有个特别聪明的地方，值得软件工程师品一下：**云并不是 Windows 的敌人，微软硬是把它变成了 Windows 的接班人。**老微软的护城河是"企业的一切都建在微软的技术栈上"。云时代，企业的一切开始搬到别人的服务器上——这本来该要了微软的命。但微软反过来说：那就让"别人的服务器"也是我的。你多年来买的 Windows Server、SQL 数据库、Office、活动目录（管员工账号的那套），我让它们在 Azure 上无缝衔接。于是那些被 Windows 生态绑了二十年的大企业，搬家时的第一选择自然是 Azure。**它把旧护城河当成了通往新护城河的桥。**

## 为什么别人做不到，它做到了

现在回到本章那个核心问题：逃出创新者窘境这么难，微软凭什么？把它的做法抽象出来，其实是几条可复用的"反窘境"原则：

- **承认新平台会杀死旧生意，而不是指望旧生意能扛住。**大多数巨头输在这一步——它们心里假设旧摇钱树还能撑很久，于是把新业务当副业养。微软的赌法反过来：假设 Windows 一定会衰落，那就在它还值钱的时候，用它换一个新地基。
- **让新旧业务不打架，而是新业务吃着旧业务的奶长大。**Office 上云、Windows 生态导流给 Azure——不是"新部门 vs 老部门抢预算"的内战，而是让老资产给新平台输血。窘境的本质是内部利益冲突，微软的解法是把冲突改造成协同。
- **忍得住财报难看。**自我蚕食一定伴随短期数字下滑。多数上市公司 CEO 熬不过那几个季度的股价压力。微软赌对了一件事：投资人要的是未来的现金流，只要你能把"我为什么现在难看"讲清楚，市场会给你时间。

大白话

一句话总结微软的翻身：它没有等到那座旧桥彻底没人走才动手，而是趁桥还挤满人、还在疯狂收钱的时候，把收上来的钱全砸去下游修新桥、买轮渡，然后引导桥上的人一个个走过去。等旧桥真的空了，公司早就靠新桥新船活得更好了。

这就是为什么微软是**极少数的反例**——不是因为它聪明，而是因为它敢在自己最强的时候，主动示弱、主动放手、主动把命运押到一个还在赔钱的东西上。绝大多数巨头做不到这一点，不是能力问题，是勇气和结构的问题：手里的摇钱树太香，谁都舍不得先砍。

但请注意，微软能"转身"，有个隐含的前提：它转身的目标——云和订阅——终究还是**它自己能掌控的地基**。它是从一块自己的地，跳到另一块自己的地。可如果有一天，你的整个生意压根就不长在自己的地上，而是长在别人的平台、别人的规则、别人的分发渠道上呢？那时候你连"转身"的资格都没有——人家一纸政策改动，就能让你的全部价值一夜蒸发。下一章，我们去看那些把身家性命交到别人手里的公司，是怎么死的。

# 第十二章 · 一家公司的所有价值都在别处

上一章我们看到一个罕见的好结局：微软这样的巨人，硬是转过身来，主动砸掉自己的旧地基，跳上云和订阅的新船，完成了第二次腾飞。那是"我踩在自己的地基上，所以我能拆能改"的故事。

这一章讲反过来的、也更常见的故事：**你踩的地基，是别人家的。**

先抛一个反直觉的现象。有一家做游戏的公司，用户数以亿计，产品火到全美国人茶余饭后都在聊，收入一年几十亿人民币量级，上市那天风光无两。然后，短短一两年，市值蒸发了八成以上。它没被对手打败，没出丑闻，没资金链断裂。它是被一个它**惹不起、也躲不开**的"房东"——Facebook——悄悄改了几条规矩，就打回了原形。这家公司叫 Zynga（一家靠"开心农场"那类社交游戏起家的公司，你可能玩过它的《FarmVille／农场小镇》）。

问题来了：一家有几亿用户、有真金白银收入的公司，护城河怎么会被"改几条规矩"就抽干？这一章要把这个机制拆开给你看。

## 先搞清楚"分发"这两个字值多少钱

软件公司的命，说到底是两件事：**做出东西**，和**把东西送到人面前**。前面几章我们死磕的都是第一件——代码、格式、网络效应、创新速度。但有一股力量，一直藏在幕后：分发，就是"怎么把你的软件塞进用户手机、怎么让用户找到你、怎么向用户收到钱"这条通道。

在 PC 时代，分发这事儿是**开放**的。你写个 Windows 软件，刻成光盘，或者挂到网上让人下载，谁也拦不住你。Windows 是地基没错，但它不站在你和你的用户中间收过路费。

手机时代变了。你想把 App 装进一台 iPhone，**只有一条路**：苹果的 App Store（苹果开的、也是唯一的官方应用商店）。这条路是苹果修的、苹果管的、苹果随时能设卡的。这就引出了这一章真正的主角机制。

大白话

换句话说：以前分发是一片公海，谁都能开船。现在分发是一条别人挖的运河，你的货必须从这儿过，而运河两头的闸门，攥在别人手里。你越赚钱，闸门费越显眼。

## 那 30% 到底是什么？是税，不是价格

你在 App Store 里卖一个 App、或者卖 App 里的道具、订阅，苹果长期抽走其中的 30%（后来对小开发者、对第二年的订阅有降到 15% 的情况，但"苹果拿走一大块"这件事没变）。

很多人第一反应是："30% 佣金嘛，贵是贵，忍忍。"这么想，就没看懂。关键不在数字大小，在于它的**性质**：

- 这不是一次性的进场费，是**对你每一笔收入永久抽成**。你越成功，它拿得越多，而且你没有议价权。
- 它不是你和用户之间的价格，是**横在你和用户之间的一道墙**。用户的钱先进苹果口袋，苹果再分给你。谁是这段关系的主人，一目了然。
- 最狠的是：这条规则**可以随时改**。今天 30%，明天苹果说"你这类 App 得走我们的内购、不许在 App 里告诉用户去网页上更便宜地买"，你就得照办。

所以真正的问题不是"抽成高"，而是**定规则的笔不在你手上**。你以为你在经营一家公司，其实你是在别人的地皮上开店，租约的条款，房东单方面就能改。

## 回到 Zynga：把命交给一个平台是什么下场

Zynga 早年几乎全部的用户和收入，都来自 Facebook 这一个平台。它靠两样东西活着，而这两样都是 Facebook 给的：

1. **免费的病毒式传播**：你在农场里收了菜，游戏自动往你朋友的动态里发一条"某某收获了…"，你朋友看见就点进来了。这等于 Facebook 借它自己几亿人的社交网络，给 Zynga 白送用户。这就是前面讲过的网络效应，只不过这张网不是 Zynga 的，是**租来的**。
2. **平台内的支付**：玩家买虚拟道具，走 Facebook 的支付通道，Facebook 从中抽成（也是约 30% 这个量级）。

然后 Facebook 做了两件对它自己完全合理、对 Zynga 却是釜底抽薪的事：

第一，**关掉那根白送用户的水龙头**。Facebook 发现满屏都是"某某在农场收了菜"的骚扰式消息，用户烦。它一改推送规则、一收紧动态里的游戏消息，Zynga 那台靠病毒传播转起来的获客机器，转速就掉下来了。它不用封杀你，只要**不再免费帮你拉人**，你就得自己花钱买用户——而买来的用户远比白送的贵。

第二，Facebook 自己的重心从网页挪到了手机。用户跑去手机上了，而在手机上，Zynga 又得重新面对苹果和谷歌这两个新房东，重新交一遍"过路费"。

> 大白话
>
> Zynga 的护城河，仔细一看，全是**别人的水**：用户是 Facebook 的、传播渠道是 Facebook 的、支付是 Facebook 的。它自己真正拥有的，只有几行游戏代码。房东把水一断，河床立刻见底。它的"所有价值都在别处"——都在那个它控制不了的平台的规则里。

## 这不是运气不好，是结构决定的

你可能会说：Zynga 是不是太蠢，太依赖一个平台了？但请注意，几乎**每一次**都是这个剧本，换的只是主角：

- 靠搜索引擎排名活着的内容站，谷歌一次算法改版（悄悄改了"哪些网页排前面"的评分规则），第二天流量腰斩，公司当月就发不出工资。它没做错任何事，是评分标准变了。
- 靠某电商平台流量活着的卖家，平台一改推荐规则、或者自己下场卖同类货，依附者瞬间失血。
- 靠某社交平台开放接口活着的第三方工具，平台哪天觉得"这块我自己做"，直接把接口一关（断供，就是平台切断你调用它数据和功能的权限），你的产品当场变砖。Twitter 当年对第三方客户端就干过这事。

看出共同点了吗？这些公司都做出了真东西、有真用户、有真收入——但它们**没有拥有那条通往用户的路**。护城河看着满满的，其实每一滴水都是上游放下来的。上游的闸门一动，护城河的深浅就跟你的努力、你的产品好坏，**一点关系都没有了**。

这就是这一章要钉死的那句话：**当你整个生意都跑在别人的分发、平台、规则之上，你的护城河不由你的能力决定，而由平台的一念之间决定，可以被一次政策改动清零。**

### 那为什么平台总是干得出这种事？

因为对平台来说，这甚至不是"使坏"，是**顺理成章**。平台看着上面某个 App 特别赚钱、特别重要，会本能地想两件事：一是"这块利润我为什么不自己吃"，二是"这么依赖我，我多抽点它也跑不掉"。平台越强，站在它地基上的公司议价权越弱——这跟第五章微软用捆绑绞杀 Netscape 是同一股力量的升级版：那时是"我送你免费的枪逼你退场"，现在是"你必须在我的地盘上开枪，弹药和规则都归我"。

## 那到底该怎么办？

没有免费的解药，但机制看懂了，方向就清楚了。真正能扛的公司，都在做同一件事：**把命根子从平台手里抢回来一部分**。

- **别把用户关系押给平台**：想尽办法拿到用户能直接触达你的东西——自己的账号、邮箱、独立入口。让用户是"你的用户"，而不是"平台的用户里恰好用了你"。
- **别只站一块地基**：多平台分散，任何一个房东翻脸，你不至于当场归零。
- **拥有一样平台拿不走的东西**：可能是独有的数据、可能是极强的品牌、可能是用户切换的高成本。这恰恰是下一章的主线。

因为你有没有发现——当分发被平台垄断、代码又能被开源填平、格式的锁也越来越松之后，护城河这东西并没有消失，它只是**搬家**了。它搬去了一个平台一时半会儿抢不走、竞争对手也复制不了的地方：**数据**。谁的数据多，谁的产品就更好，产品更好又让谁的数据更多——这个自己会滚雪球的循环，成了新的锁。下一章，我们就去看这把新锁是怎么铸成的。

# 第十三章 · 数据成了新的锁

上一章我们看到，当你的整个生意都跑在别人的地基上——别人的商店、别人的分发、别人的规则——一次抽成调整、一次算法改版，就能把你几年攒下的价值一夜清零。那一章讲的是"你依附于谁"。这一章要问一个更狠的问题：**如果护城河本身，正在从"你写的代码"迁移到"你手里攒的数据"，那到底谁才真正握着开关？**

先抛一个反直觉的现象。搜索引擎的算法，几乎全世界都知道大致原理；推荐系统的论文，公开发表得满地都是；连训练大模型的方法，也早就不是秘密。按理说，代码和方法一旦不再稀缺，护城河就该填平了——第七章我们不是刚讲过开源怎么把"代码所有权"这道墙推倒的吗？可现实是，Google 搜索、YouTube 推荐、头部大模型，反而越做越难被追上。方法人人都懂，为什么后来者还是打不过？

答案藏在一个词里：数据飞轮——不是"数据多所以厉害"这么简单，而是一个会自己转起来、越转越快的循环。

## 飞轮到底是个什么轮子

拆开看，它就四步，绕成一个圈：

1. 你有更多用户在用你的产品；
2. 他们的每一次点击、跳过、停留、纠错，都变成了数据；
3. 这些数据喂回去，让你的产品变得更准、更好用；
4. 产品更好，就吸引来更多用户——回到第一步，只是圈更大了。

关键在于：**这个圈每转一圈，就把你和对手的差距又拉开一点。**对手就算今天把你的代码一行不差抄走，他也拿不到你这几年里几十亿次点击攒出来的那份"经验"。代码是可以复制的，数据不是——尤其是那种"用户在真实场景里用你的产品才会产生"的数据。

大白话

换个说法：以前打软件仗，比的是"谁的产品做得好"。现在比的是"谁的产品被用得多"——因为被用得越多，产品就自动变得越好。这是一种复利。你不是在跟对手比一次考试的分数，你是在比谁的雪球已经滚了更久。

### 为什么这种护城河比前几章的都更硬

回头数一数前面讲过的护城河：代码所有权（第一章），被开源填平了；文件格式的绑定（第三章），标准一开放就松动；连网络效应（第四章），也可能被一个更好用的新平台掀翻。这些护城河都有个共同弱点——它们是**静态的**。你今天挖好一条壕沟，它就在那儿，对手迟早能想办法填、想办法绕。

数据飞轮不一样，它是**动态的**。它不是一条挖好的壕沟，而是一台一直在往外扩地盘的机器。对手追赶你的同时，你自己还在加速。这就是为什么后来者常常有种绝望感：不是差距大，而是差距**在你追的时候还在扩大**。

## Google 是第一个把这台机器造出来的公司

第六章讲过 Google 靠广告赚钱，把"用户即货"这件事做成了。但那一章讲的是**商业模式**怎么转向广告。这一章要补上另一半：Google 真正的护城河，其实是那台在广告底下悄悄转动的数据飞轮。

想想它早年那个不起眼的功能——"你是不是想搜……"（Did you mean，就是你打错字时它猜你真正想搜什么）。它怎么知道你拼错了？因为在你之前，已经有数以百万计的人打错过同样的字，然后又改对重搜了一遍。Google 看着这几百万次"打错—改对"，就学会了正确拼法。**你每一次犯错并修正，都在替下一个人把答案调得更准。**用的人越多，它就越聪明；越聪明，用的人就越多。这台飞轮，从二十多年前就开始转了。

后来 YouTube 的推荐、地图的实时路况，走的都是同一条路。推荐系统——就是那个决定"下一个自动播给你看什么"的东西——它的好坏，几乎全押在数据量上。你没法靠一个更聪明的算法凭空追上一个看过几十亿小时观看记录的系统，因为它见过的人类偏好，比你多太多了。算法是次要的，喂进去的东西才是主角。

## 到了大模型时代，飞轮换了个更大的轮子

轮到今天。大模型（就是 ChatGPT 那类能对话、能写东西的 AI）表面上看是"算法的胜利"，好像谁的模型结构更巧谁就赢。但真相更接近前面那台老飞轮——只是喂料的方式变了几层。

这里要小心分清三种不同的"数据"，因为很多人一锅炖，就看不清护城河到底在哪：

- **预训练用的海量文本**：从公开互联网上扒来的。这部分其实**不太构成独占护城河**——大家能扒的地方大同小异，这也是为什么好几家公司能做出水平接近的模型。
- **人类反馈数据**：RLHF，说人话就是"让人给 AI 的回答打分、挑毛病，再拿这些评价去调教它"。谁有更多真人在持续帮它纠偏，谁的模型就更好用、更不容易胡说。这份数据是花钱、花时间、花用户量才攒得出来的。
- **真实使用数据**：几亿人每天拿它干什么、在哪句话上点了"重来"、哪个回答被复制走了——这一份，只有已经拥有大量用户的产品才拿得到。**这才是新的锁。**

> 大白话
>
> 所以大模型时代的竞争，第一步比的是谁能扒到、买到、算得动海量数据（这一步有钱就能追）；但真正拉开长期差距的，是产品上线之后那台飞轮——谁的用户多，谁就拿到更多真实反馈，模型就更好用，于是用户更多。绕回来了，又是同一个圈。

这也解释了一个乍看奇怪的现象：为什么那些手里早就攥着海量用户和数据的巨头，在这一轮里既紧张又占便宜。紧张，是因为新玩家可能造出比他们更好的模型（这是下一章要正面撞的事）；占便宜，是因为一旦模型接进他们那些几亿人天天在用的产品里，飞轮当场就能转起来——他们不缺喂料的入口。

## 飞轮不是永动机——它也会卡住、也会反噬

别把数据飞轮当成一劳永逸的护城河，那就又掉回"正确的废话"里了。它有几个真实的裂缝，值得记住：

**一是数据会有天花板。**当模型已经好到 99 分，用户反馈能带来的提升越来越小——飞轮转得动，但每圈推着雪球长大的那点雪，越来越薄。护城河还在，但不再*加宽*。

**二是飞轮可能反着转。**如果产品变差、用户流失，数据就变少、质量下降，产品继续变差——同一台机器，倒着转就是加速下坠。护城河可以蒸发，正是因为让它成立的那个循环，本身是脆弱的。这不就是全书开头那句话吗：**软件走向垄断，却又天生脆弱。**

**三是数据的所有权正在被重新争夺。**当"谁有数据谁赢"成了共识，那些生产数据的一方——写内容的人、被拿去训练的网站、贡献反馈的用户——开始追问：凭什么是你在攒这份复利？这场关于"数据到底归谁"的拉锯，才刚刚开始。

看清了这一层，你就摸到了这本书的一条暗线：护城河从没消失，它只是不停地*换地方*——从代码，到格式，到网络，再到今天的数据与规模。每换一个地方，上一批守着旧壕沟的公司就会突然发现，自己的墙不知何时已经不挡水了。

而现在，地基本身又要动了。数据是这一轮的锁，可锁装在门上，门装在墙上，墙立在地基上——如果整块地基被换成一种全新的东西呢？下一章，我们就去看这场几十年一遇的平台大更迭：AI 不只是又一个产品，它可能是把前面每一股力量——平台绑定、网络效应、创新者的窘境、模式的地心引力——同时重新洗一遍牌的那件事。

# 第十四章 · 平台又要换了

上一章我们看到，护城河从代码、格式、网络，一路搬家搬到了"数据"上——谁数据多，谁产品好，产品好又带来更多数据，滚成一个越转越快的飞轮。故事讲到这里，你可能以为软件公司的命运格局已经定了：谷歌坐拥全世界的搜索数据，微软手握全世界的办公文档，数据复利一旦滚起来，后来者根本追不上。

可是从 2022 年底开始，一件很反直觉的事发生了：一家当时只有几百人、连正经收入模型都还没想清楚的小公司，几个月内就让市值上万亿的巨头们集体进入"红色警戒"。谷歌内部发了内部拉响警报的通知，微软则干脆掏出上百亿美元把这家小公司抱进怀里。这家公司叫 OpenAI——一个做 AI 大模型的实验室，它的产品 ChatGPT 是一个你打字它就用人话回答你的聊天框。

问题来了：数据飞轮不是号称"赢家通吃、追不上"吗？为什么巨头们反而怕成这样？这一章要回答的，就是这个：**当平台本身要换代的时候，之前所有让巨头强大的力量，会在同一瞬间集体反转，变成拖住它们的枷锁。**

## 什么叫"平台又要换了"

先把"平台更迭"这个词拆开。第二章讲过 平台——就是别人在你身上盖房子的那块地基，比如 Windows 之于当年所有的 PC 软件。历史上每隔十几二十年，地基会整个换一次：从大型主机换到 PC，从 PC 换到浏览器，从浏览器换到手机。每换一次，上一代地基上的霸主就要重新证明自己——很多没证明成功，比如死在平台更迭里的 Lotus、WordPerfect。

现在轮到第五次换地基了。这一次的新地基，是"你用大白话下命令，机器直接给你结果"这种交互方式。过去几十年我们跟电脑打交道，本质都是"人去学机器的规矩"：点这个菜单、填那个表格、记住这条命令。而大模型第一次让"机器来迁就人的话"成了可能。这不是多了一个 App，而是**人机之间那层界面，整个换了一种材质**。界面一换，谁站在用户和答案中间，就要重新洗牌。

大白话

翻译成人话：以前你想知道点什么，得自己去谷歌搜一堆蓝色链接，再自己点进去读、自己拼答案。现在你直接问一句，机器把答案端到你面前。中间那个"你自己动手翻链接"的环节，正是谷歌卖广告的地方。地基换了，收费站可能就没了。

## 为什么巨头的"强"会变成"弱"

这才是本章最要命的地方。前面几章讲过的每一股力量，单看都是护城河，可一到换代关口，它们会同时翻面。

### 第一面：创新者的窘境

第八章讲过 创新者的窘境——不是大公司笨，而是它太成功，成功本身让它没法转身。谷歌一年从搜索广告里赚几百亿甚至上千亿美元，这门生意的前提是"用户多点几次、多看几个链接、多看几条广告"。而一个直接吐答案的 AI，恰恰让用户不用点链接了。

你让谷歌的产品经理去做一个"让用户不再需要点广告"的东西，他理性上做不出来——那等于亲手拆自己的钱袋子。所以 2022 年谷歌其实早就有能力做出类似的模型（它的研究员甚至是"Transformer"这套技术的发明人），但它把这些东西压在实验室里，迟迟不敢正式端出来。**不是不能，是不敢。**OpenAI 没有这个包袱，光脚的不怕穿鞋的，一脚就把门踹开了。

### 第二面：模式的地心引力

第六章讲过，赚钱模式像地心引力，会把公司的每一个决定往一个方向拽。谷歌被"广告"这个引力拽了二十年，它的一切——排序、界面、甚至什么算"好答案"——都是为了多卖广告。现在冒出一种新模式：用户直接订阅（每月付固定的钱换服务，第九章讲过）大模型的能力，或者开发者按调用次数付费。这个新引力场里没有广告的位置。

一家习惯了在 A 引力场里飞的公司，很难在 B 引力场里重新学会飞。它每往新模式挪一步，都在跟自己二十年的肌肉记忆打架。

### 第三面：网络效应与数据飞轮反倒帮了新人

最反直觉的是这个。上一章说数据飞轮让巨头追不上，可换代时它有个漏洞：**旧飞轮攒的是旧地基上的数据。**谷歌攒了海量"用户点了哪个链接"的数据，这对排搜索结果极有用，但对"怎么把话说得像人"帮助有限。新地基需要的是新数据——海量的文本、还有海量真人跟 AI 对话时的反馈。

而这种新数据，谁先把产品端出去、谁先有几亿人天天在上面聊天，谁就先攒。OpenAI 靠先发，几个月内就攒起了一个巨头暂时没有的新飞轮。数据复利这把武器，这一次先握在了新人手里。

## 那微软呢——它凭什么这次没慌

同样是巨头，微软的反应完全不同：它不但不怕，还主动往火里跳。为什么？

因为微软早就吃过一次大亏，也在第十一章里完成过一次罕见的自我革命——它已经把身家从"卖 Windows"挪到了"卖云"。云就是第十章讲的，把计算能力当水电一样按量出租。而大模型这东西，训练和运行都要吞掉天文数字的计算资源。也就是说，**不管最后哪家 AI 公司赢，它们都得在别人的云上烧钱**——微软相当于卖水给所有淘金的人。

所以微软的算盘是：投资 OpenAI，既把最强的模型绑进自己的云和 Office 里（让 AI 帮你写邮件、做表格），又保证无论战局怎么变，算力这层的税它照收。它没有被"旧模式"绑住，因为它几年前就已经忍痛把自己从旧模式里拔了出来。这正是第十一章那个稀有反例的续集：**能逃出创新者窘境的公司，靠的不是运气，是提前一步、主动把自己旧护城河的水放掉。**

大白话

翻译成人话：同一次浪打过来，谷歌怕呛水，因为它整个身子还泡在旧生意里；微软不怕，因为它早几年就爬到"卖水电"这块高地上了，浪越大，用水电的人越多，它反而越赚。你站在哪块地基上，决定了同一件事对你是灾难还是机会。

## 但别急着给新人发奖杯

讲到这你可能觉得：好，新人赢了。可这本书从第一章就在提醒一件事——软件的垄断天生脆弱。新地基上的护城河，会不会也一夜蒸发？

看几个隐患就懂了。第一，最好的模型今天领先，明天可能被追平：训练大模型的方法是公开的，钱和算力谁都能砸，先发的技术优势不像想象中那么稳。第二，第七章讲的开源——把代码免费公开让所有人用——正在 AI 领域重演，一堆免费开放的模型正在填平"模型所有权"这条护城河，就像当年 Linux 填平操作系统那样。第三，也是最狠的：真正掌握新地基的，未必是做模型的人，而可能是掌握分发的人——手机系统、浏览器、操作系统。谁能把 AI 直接塞进几十亿人每天都在用的入口，谁就掐住了咽喉。这正是第十二章的老问题：**你的价值如果跑在别人的分发渠道上，一次改动就能把你清零。**

所以这一章真正的洞见不是"AI 来了、巨头要完"，也不是"新人一定赢"。而是：**换代关口，是这本书里讲过的每一股力量——平台绑定、网络效应、创新者窘境、模式引力、开源、分发权——同时上桌重新发牌的时刻。**谁在死守旧牌，谁在下注新牌，几年之内就见分晓。有的巨头会像谷歌一样惊出一身冷汗然后奋力掉头，有的会像微软一样早有准备顺势而起，也一定会有一批当红公司，在这次换代里悄无声息地沉下去——就像每一次换代都发生过的那样。

写到这里，全书的十四股力量已经一股不落地登过场了。它们看起来五花八门：有的关于成本，有的关于网络，有的关于数据，有的关于平台。但如果把它们摊在一张纸上，你会发现它们其实在讲同一件事——一套关于"软件公司为什么长盛、为什么速朽"的力学。下一章，我们就把这些力量收成一张受力图，从 IBM 一直看到 OpenAI，回答我们在第一页就问出的那个问题。

# 第十五章 · 兴衰的力学

前面十四章，我们像拆钟表一样，一个齿轮一个齿轮地看软件公司为什么活、为什么死。现在钟表拆完了，零件摊了一桌子。这一章不加新零件，只干一件事：把它们重新装回去，让你看清这台"兴衰机器"整体是怎么转的。

我先抛一个反直觉的事实。**过去五十年，几乎每一家倒下的软件巨头，倒下的时候都还在赚钱、还在增长、产品还在变好。**诺基亚被 iPhone 掀翻那年是它出货量的历史最高峰；柯达发明了数码相机却死于数码相机；WordPerfect 被微软干掉时，它的文字处理器公认比 Word 好用。这就怪了——按常理，产品更好、客户更多、钱更多，应该更安全才对。可现实是，护城河最深的时候，往往就是它最脆的时候。

要理解这个悖论，你得换一个视角：别把公司看成一个"东西"，把它看成一个"受力物体"。

## 一张受力图：五股力量，同时拉着每一家公司

物理里分析一个物体，你不看它长什么样，你画一张受力图——箭头，方向，大小。软件公司也一样。全书讲的五股力量，其实就是同时作用在每家公司身上的五个箭头：

- 平台更迭：你脚下那块地基（大型机 → PC → 浏览器 → 手机 → 云 → AI）会整块换掉。这是一股竖直向下的重力，永远在，只是平时你感觉不到。
- 绑定与解绑：一股把客户"粘"在你身上的力（数据搬不走、格式打不开、员工换不起），和一股相反的、想把这层胶溶解掉的力（开源、开放格式、云上一键迁移）。这两股力天天在拔河。
- 网络效应：用的人越多，产品对下一个人越值钱。这股力赢家通吃，但它有个魔鬼细节——它不认"你是谁"，只认"哪儿人多"。人潮换个方向，它立刻反过来推你。
- 创新者的窘境：你现在赚钱的生意，会亲手拦住你去做那个未来会吃掉你的新生意。这不是老板蠢，是"越理性越会往坑里走"的力——赚钱的部门声音大，亏钱的新东西没人护。
- 模式的地心引力：钱怎么收（买断 → 月租 → 广告 → 抽成）会悄悄决定你把整家公司往哪个方向长。这股力最隐蔽，因为它伪装成"商业常识"。

大白话

说人话：一家公司此刻是涨是跌，不取决于它"好不好"，取决于这五个箭头加起来指向哪。产品再好，也只是其中一个箭头，而且往往是最短的那个。

## 为什么"越强越脆"：护城河和陷阱是同一样东西

现在用这张图，来解那个悖论。关键洞见是：**让你强的那股力，和会杀死你的那股力，常常是同一股，只是换了个方向。**

拿绑定来说。锁住客户的护城河，是让别人进不来；可平台一换代，这条护城河就变成一条把你自己焊死在旧船上的锁链。WordPerfect 靠一整套复杂快捷键把用户绑成了肌肉记忆——这是它的护城河。可当地基从 DOS 换到 Windows，它舍不得推翻这套老逻辑去重做，微软的 Word 却是为 Windows 从头长出来的。绑住客户的那根绳，最后把它自己勒死了。

网络效应更狠。它是正反馈：人多 → 更值钱 → 更多人来。但正反馈没有方向偏好，它只是个放大器。风往你这边吹时它帮你滚雪球，风一转向，它同样帮你的对手滚雪球，而且滚得一样快。这就是为什么网络效应型的公司，崩起来是"一夜蒸发"而不是慢慢衰退——放大器不会温柔地关机。

所以护城河的深度，某种意义上就是塌方时的高度。你把自己修得越牢固、越依赖某一块地基、某一种收钱方式、某一群人的聚集，你对那股力的敏感度就越高。平时它托着你，变天时它砸着你。是同一只手。

## 那为什么还有人能长盛几十年?

如果每股力量都这么无情，按理说没有公司能活过一次平台更迭。但确实有——IBM 活过了大型机的黄昏，微软活过了 PC 时代的落幕。他们凭什么?

答案不是"他们护城河更深"，恰恰相反：**是他们敢在护城河还满水的时候，亲手把它放空，去挖一条新的。**

第十一章那个微软的例子值得再看一眼，因为它是整本书里最稀有的动作。自我蚕食——就是主动砸掉自己正在赚大钱的产品，为了抢在别人砸掉它之前。当年 Windows 是微软的命

根子，几乎所有决策都得"对 Windows 好"。新掌门人做的事，本质上是把公司的重心从"守住脚下这块地基"改成"押注下一块地基"——云和订阅。这等于承认：我现在最赚钱的东西，不是我的未来。

用受力图看，这些幸存者干的都是同一件反直觉的事：他们不跟力量对抗，他们换立足点。既然重力（平台更迭）永远把你往下拽，那唯一的活法不是"站得更稳"，而是"在旧地基塌之前，脚已经迈到新地基上"。守旧的人在加固城墙，幸存的人在搬家。

## 一套能自己用的"看兴衰"工具

把上面的东西收成几个你下次看到任何一家公司都能直接套的问题。这才是这本书想留给你的东西——不是一堆故事，是一副能自己戴的眼镜：

1. **它站在谁的地基上？** 它的生死是不是攥在别人手里（App Store 的抽成、某个平台的算法、某家云的账单）？地基方一句话能不能让它的价值归零？（第二、十、十二章）
2. **它的护城河是"胶"还是"墙"？** 是靠客户搬不走（绑定），还是靠人多（网络效应），还是靠数据滚雪球（第十三章）？然后追问：哪一种力量正在溶解这道墙？开源、开放标准、还是新平台？
3. **它现在赚的钱，会不会拦住它的未来？** 那个可能吃掉它的新东西，是不是正好威胁它最赚钱的部门？如果是，它八成不会主动去做——这就是窘境的味道。
4. **钱怎么收的，正在把它往哪儿长？** 收租的会拼命防客户流失，卖广告的会把用户变成货，抽成的会拼命把人锁在自己的集市里。看它收钱的方式，就能猜到它三年后长成什么样。
5. **地基要换了吗？** 如果一次平台更迭正在发生，前面四问全部要重算一遍——因为更迭会同时激活所有力量。

现在回到全书那个贯穿始终的问题：为什么有的软件公司长盛几十年，有的巅峰即坠？

大白话

不是因为谁更聪明、谁产品更好、谁更努力。是因为软件这个行业，地基本身每隔十几年就要整块换一次；而绑定、网络效应、赚钱模式这些让你强大的力量，全都对"地基"极其敏感——地基不动时它们是护城河，地基一动它们就是陷阱。长盛的那几家，无一例外，是在自己最风光、最不需要改变的时候，选择了搬家。坠落的那些，是在同样风光的时候，选择了加固。

从 1969 年 IBM 被迫把软件和硬件拆开卖，到今天 OpenAI 和在位巨头在 AI 这块新地基上正面相撞——五十多年，公司换了一茬又一茬，但那张受力图上的箭头，一根都没变。变的只是它们指向谁。

而此刻，我们正站在这本书里最大的一次地基更迭上。AI 不是又一个功能，它是又一块要整体替换的地基（第十四章）。这意味着前面这套工具，第一次可以被你用在"正在发生"的历史上，而不是复盘。所以别把这五个问题当成读后感——把它们记住。因为接下来几年，你会亲眼看到一批今天看起来坚不可摧的公司，用书里一模一样的方式，把护城河变成陷阱；也会看到极少数几家，在所有人都以为它们赢麻了的时候，悄悄开始搬家。到时候，你会认得那股力。