每 日 必 读 · 典 范 文 · 之 一

# 为什么非得停下来

盘点要关门，修路要封路，清理内存要卡一下，而人每天要交出七八个小时。最顺口的解释是「维护和干活抢同一副硬件」——可如果只是资源问题，加倍投入就该解决它。本文认为三家成熟传统都漏了同一件事：停机不是为了腾出资源，是为了冻结判断的前提。

王德生 + Claude · 约 1.6 万字 · 17 页 · 三种阅读方式 · 发表于2026年8月1日

清点一百件货，力气可以忽略不计——可只要收发不停，你就永远得不到一份对的数。而冲刷一条管道里的沉积极耗资源，却可以一边用一边冲。**这两个例子放在一起，「停机是为了腾出资源」这个解释就塌了。** 三家成熟传统在这件事上立场互相取消：睡眠研究说那段离线不可取消；并发内存回收三十年的纲领就是消灭停顿，把必须暂停的窗口从几秒压到亚毫秒；会计取消了停业盘点，却坚决保留截止日，而它自己给的两个理由互相推不出。本文提出的判断是：**一项维护之所以必须停机，是因为它所依据的那个判断会被在线活动推翻——停机是为了冻结判断的前提**；由此得到一条可判定的判据：看那个判据是不是**单调**的（一旦成立就不会被逆转）。这条判据不是思辨，工程界已经算了三十年，只是把它写成了自己的局部事实。文末给十处跨领域读数、两处反过来迫使它让步的地方、六条能推翻它的条件（最便宜的一条只需翻一份公开的功能演进史，另一条只需一间仓库和一天），以及一条写死到 2032 年的赌注。

## 引言

这篇文章要处理的现象随处可见，却几乎从不被当作一个需要解释的对象：**有些维护必须让被维护的东西先停下来**。

它值得现在问，有两个理由。其一，"减少停机"是过去二十年里几乎所有系统改造的共同目标，从工厂到医院到数据中心，而这些改造的成败极不一致——有的一举成功，有的反复失败，还有的成功之后过几年又回到原点。一个反复被尝试却结果两极的地方，通常说明大家改的不是主变量。其二，有一支行当已经把这件事算到了极精确的程度，只是把答案写成了自己的术语；把它搬出来，另外两支争了很久的事情可以当场安顿。

本文立的判断只有一条：**决定一项维护能不能不停机的，是它所依据的判断会不会被在线活动推翻，而不是它耗多少资源**。

路线图如下：第二到第四章逐一交代三支说到了哪一步、又在哪里停下；第五章把矛盾写死；第六章给出变量的定义与三个可以直接问的问题；第七章加上第二根轴，得到四格与一条等价关系；第八章给四格各找真实标本；第九章摆出三条已经被算清楚的读数；第十章说明三支为什么各自推不出这一条；第十一章在十个别的领域里兑现；第十二章报告两处迫使命题让步的地方；第十三章与最接近的六种说法逐条分界；第十四章交出不适用的范围；第十五章写清按它设计干预会怎样出事；第十六章给出证伪清单；第十七章回答"为什么这件事一直没被看见"；第十八章处理本文自己也是标本这件事；末章是一条写死日期的赌注。

## 一、三家都在处理同一个问题，而它们的处方互相取消

先摆一件所有人都知道、却很少被当成问题的事：**有些维护必须让被维护的东西先停下来**。

盘点的时候商店要关门，清理内存的时候程序要卡一下，修路的时候要封路，做手术的地方要固定住不能动。而人自己每天要交出七八个小时，在那段时间里几乎不接收任何新东西。

为什么？最顺口的答案是"资源被占用了"——维护要用的力气、机器、注意力，跟平时干活要用的是同一份。这个答案听起来无懈可击，也确实解释了一部分。但它有一个明显的漏洞：**如果只是资源问题，那么加倍投入就该解决它**。而现实里，有些停顿无论投多少资源都取消不掉，另一些则在某次不起眼的技术改进之后悄悄消失了。

对这件事，有三支互不相干的传统给出了各自的说法，而它们的处方互相取消。

**第一支来自睡眠研究**。 它的立场是：那段离线不可取消。至于离线时到底在做什么，这一支内部还在打——一派认为整晚的主要工作是把白天被普遍加强的连接**整体调弱**，好腾出第二天的余量；一派认为主要工作恰恰相反，是**选择性地加强**某些痕迹，把它们从容易被冲掉的地方搬到不容易被冲掉的地方；还有一派把功能整个挪出信息层，认为那段时间主要在做**代谢废物的冲刷**，脑组织间隙在睡眠期间明显扩张，清除效率随之提高。三派吵得很凶，但在一件事上高度一致：**这些工作必须在离线时做，而且离线不能被取消**。

**第二支来自计算机系统，而它的整个研究纲领就是取消离线**。 早期的内存回收要把程序整个停住，这被视为不可接受的缺陷；此后三十年的工作，几乎全部投在"让回收与运行同时进行"上，一代代把那个必须暂停的窗口从几秒压到几毫秒、再压到亚毫秒。这一支的默认立场是：**停顿是工程缺陷，是可以被消灭的**。

**第三支来自会计与内控，而它给的答案是"一半一半"**。 传统的做法是停业盘点：关门、封存、逐件清点。后来这被循环盘点和持续审计逐步替代——不再关门，改成分区轮盘、随时抽查。但同一支传统同时坚持另一件不肯让步的事：**截止日不能取消**。到了某个时点，账必须被人为地划一刀，之后发生的一律算下一期。你可以取消关门，但你取消不了那一刀。

三支并排放着，麻烦就出来了。第一支说离线是根基，第二支说离线是缺陷，第三支说有的能取消有的不能——而它自己也说不清那条界线画在哪里。

本文要主张的是：**三支都在同一个地方失手。它们各自盯着停顿的一个属性——不可取消性、可优化性、可替代性——却都默认了同一件事：停顿是为了腾出资源**。

一旦把这条默认拿出来当变量，答案的形状就变了。本文的承重判断是一句话：

> **一项维护之所以必须让系统停下来，不是因为它耗资源，也不是因为它与在线活动争用同一副硬件，而是因为它所依据的那个判断会被在线活动推翻。停机是为了冻结判断的前提，不是为了腾出资源**。

而这句话给出了一条可判定的判据，第六章会把它写清楚：**看那个判断所依据的属性是不是单调的**——一旦成立，就不会被在线活动逆转。

## 二、睡眠那一条说到哪一步

第一支的三派，本文只关心它们共享的那一步。

**第一派**主张：清醒时学习会让神经连接普遍变强，而变强是有代价的——能量、空间、信噪比都会被吃掉，且强度趋于饱和之后新的学习无处安放。睡眠期间发生的是一次**普遍的下调**，按比例把所有连接一起调弱；由于强的减得少、弱的被减没，相对关系被保留下来，而绝对水平回到了可以继续学习的位置。

**第二派**主张：睡眠期间发生的不是普遍下调，而是**选择性的重演**——白天形成的某些活动序列在夜间被以压缩的速度重播，重播的那些被搬到更稳固的地方去，没被重播的则逐渐消退。证据来自对同一批细胞的连续记录，以及打断特定睡眠阶段之后特定记忆的选择性受损。

**第三派**把功能整个挪出信息层。它主张睡眠期间脑组织间隙扩张、液体流动加快，代谢废物被更有效地冲走；这一支的证据链是解剖与流体的，与前两派谈的东西几乎不重叠。

三派对功能的归属互不相容，但请注意它们在方法上共有的一步：**它们全都把"必须离线"当作待解释项的背景，而不是待解释项本身**。

第一派会说：普遍下调不能在清醒时做，否则你会在下调的过程中不断加强新的连接，永远调不完。第二派会说：重演不能在清醒时做，否则重演会被当作真实的输入。第三派会说：冲刷需要间隙扩张，而清醒时的活动状态不允许。

**三句话是三种不同的理由，可它们的形状是同一个**：某样在线的活动会**破坏这项维护赖以进行的条件**。

第一派说的破坏是"边调边加"；第二派说的破坏是"分不清哪是回放哪是现实"；第三派说的破坏是"通道打不开"。

睡眠这一支说到了一件真的事：**这些工作在在线状态下做不成**。它没说到的是——做不成的原因不是同一种，而它们之间的差别恰恰决定了哪一项**有可能**被搬到线上、哪一项永远不能。把三个理由笼统地记成"离线才行"，就永远得不到这条区分。

而下一章那一支，恰恰把这条区分算清楚了——只不过它以为自己在解决一个纯粹的工程问题。

## 三、并发回收那一条说到哪一步

第二支的历史是一部"把停顿越压越短"的历史，而它在这个过程里被迫把停顿的**理由**拆开了。这一章要用的正是那次拆分。

早期的做法很直白：程序整个停住，从若干根节点出发遍历整张对象图，凡是走得到的留下，走不到的回收。停顿的长度与堆的大小成正比，几秒的暂停在当年是常态。

要把这个过程搬到"边跑边做"，立刻撞上一个问题：遍历的过程中，程序还在改指针。可能出现这样一种情况——某个对象在遍历还没走到它时被挂到了一个已经走完的位置上，于是遍历永远不会再回去看它，它被当成垃圾回收掉了，而它其实活着。这不是效率问题，这是**正确性**问题。

这一支给出的解法值得逐字看清楚，因为本文的整套判据就藏在里面。

**第一，它找到了一条可以依赖的单调性。** 有一条性质是不会被程序逆转的：**一个对象一旦变得不可达，就永远不可达**——程序没有办法把一个已经够不着的东西重新够着。因此，在一份**过时的快照**上做遍历是安全的：按遍历开始那一刻的可达关系去判定，判为活的可能实际已经死了，但判为死的一定真的死了。

**第二，它明码标出了这个安全的代价。** 那些在遍历期间死掉、却因为在快照里还活着而没被回收的对象，有一个专门的名字，叫**浮动垃圾**。它们不是错误，是这一轮故意放过的部分——**下一轮再收**。

**第三，它同时标出了哪些环节仍然必须停。** 不是所有环节都能靠这条单调性过关：扫描根节点、翻转线程栈、以及**搬动对象**的那一段，仍然需要一个所有人都不动的瞬间。原因也很清楚——一旦对象被搬走，所有指向它的旧地址就都错了，而在没有全局一致的瞬间里，你没法保证不会有人正拿着旧地址。

**第四，也是最锋利的一条：它发现停不停是速率的函数。** 有一套实现会盯着程序制造新对象的速度，如果快过回收的速度，就逐步削减程序自己的时间配额；在极端情况下配额被削到零——**也就是自动退回了停机**。

四条合起来，这一支实际上已经回答了"什么时候非停不可"，只是它把这个答案写成了自己领域的实现细节。本文要做的，是把这个答案从它出生的地方搬出来：

**一项工作能不能边跑边做，取决于它依据的判断会不会被在线活动推翻；如果不会（单调），就可以在过时的快照上做，代价是漏掉一批、下轮再收；如果会，就要么停下来，要么在每一次可能推翻它的动作上截一下。**

这一支说到了一件真的事，而且算得比谁都精确。它没说到的是：**这条判据不属于内存回收，它属于一切维护。**

## 四、盘点与截止那一条说到哪一步

第三支来自会计与内控，它的价值在于：**它是唯一一个把"取消掉的那部分"与"取消不掉的那部分"分别做成了制度的传统。**

传统做法是停业盘点：选一天关门，封存出入库，逐架逐件清点，账实核对，差异调整。这件事的理由从来不需要解释——不关门怎么点？点着点着货就走了。

后来它被两样东西替代了大半。一样是**循环盘点**：不再全场关门，改成把仓库分区，每天点一小块，一年转一圈；点某一区时只冻结那一区，其余照常收发。另一样是**持续审计**：不再等年末抽样，而是把规则写进系统，每一笔交易过账时自动比对。

这两样替代都成功了，而且成功得很彻底——今天大型零售与制造业里，全场停业盘点已经罕见。

可同一套传统在另一件事上寸步不让：**截止日**。到某个时点，账必须被划一刀：这一刀之前发生的算本期，之后的算下期。跨在刀口上的那些——货已发出而单未开、款已收到而服务未交付——必须被逐笔判定归属。审计里有一整套专门的程序对付这条线，因为它是最容易出错、也最容易被操纵的地方。

于是这一支给出了一个别处看不到的对照：**同一个行当，取消掉了关门，却取消不掉划刀。**

它自己给这条差别的理由通常是"技术进步"与"内控要求"——前者解释了为什么关门可以不要，后者解释了为什么刀必须划。但这两个理由不在同一层，也不能互相推出。为什么条码与实时系统能取消关门，却不能取消截止日？如果技术足够好，为什么不能"持续结账"，让每一秒都有一份正确的报表？

本文认为这一支已经摸到了那条界线，只是没有把它说出来。区别在这里：

**盘点依据的是"某个时点上有多少"，而这个数会被在线的收发持续推翻**——你点完 A 区去点 B 区，A 区已经变了。所以它要么关门，要么**冻结分区**——而冻结分区正是一道屏障：不关全场，只在那一小块上把推翻动作截住。

**而收入归属依据的是"这笔交易属于哪一期"，这件事一旦成立就不会被后来的交易逆转**——今天卖出去的东西，不会因为明天又卖了一件而变成明天卖的。所以它不需要停业，它需要的只是一个统一的时点，让所有人按同一份"过时的快照"来判。**截止日不是停机，它是快照。**

这一支说到了一件真的事：**有的停顿能被替代，有的不能。**　　它没说到的是这两类的分界线在哪——而分界线，就是上一章那条被工程界算清楚的性质。

## 五、三条不能同时为真

**第一对：离线是根基还是缺陷。**

睡眠那一支的整个立场建立在"这段离线不可取消"上；内存回收那一支三十年的工作就是证明"停顿可以被压到几乎没有"。两者对同一件事给出相反的定性，而且都有硬证据——一边是打断特定睡眠阶段带来的特定损害，一边是真实系统上从几秒压到亚毫秒的实测。

这一对不能靠"领域不同"打发。因为如果停顿只是资源占用，那么它在两个领域里就该是同一种东西，只是量级不同；而如果它在一个领域可以被消灭、在另一个领域不能，那就说明**它们之间有一条质的差别，而这条差别没有被任何一方说出来**。

**第二对：能不能取消，取决于什么。**

会计那一支的实践给出了一个双重结论：关门可以取消，划刀不能。而它自己给的两个理由（技术进步、内控要求）不在同一层，互相推不出，也无法预测下一项维护属于哪一类。

一个具体的追问就能把它逼住：如果技术再进步一步，能不能取消截止日？这一支答不了——它的框架里没有一个能回答"什么可以被技术取消"的变量。

**第三对：睡眠自己内部那三派，其实指向三种不同的不可取消性。**

第一派说的是"边调边加，永远调不完"——这是**生成速率**问题。第二派说的是"重演会被当成真实输入"——这是**判断被污染**的问题。第三派说的是"通道打不开"——这是真正的**物理资源**问题。

三种理由如果被笼统记成"离线才行"，就得不出任何区分；而一旦拆开，第三派那一条恰恰**不该**跟另外两条放在一起：如果冲刷只是需要通道，那么只要通道能开，它原则上可以在线做——而实测也确实显示清醒时同样有清除，只是慢。

三对矛盾指向同一个空缺。三支各自盯着停顿的一个属性——不可取消、可优化、可替代——却都默认了同一件事：**停顿是为了腾出资源**。

把这条默认拿出来当变量，三对矛盾同时松开：

- 离线之所以在一处是根基、在另一处是缺陷，不是因为生理与工程有什么本质差别，而是因为**两处维护所依据的判断，一处会被在线活动推翻，一处不会**。
- 关门之所以能取消而划刀不能，是因为**盘点依据的"有多少"会被收发推翻，而收入的期间归属不会被后来的交易逆转**。前者需要冻结或屏障，后者只需要一份大家公认的过时快照。
- 睡眠那三派之所以吵不出结果，是因为他们把三**种不同的不可取消性**混成了一种。

下一章给这个变量下定义，并说清怎么判。

## 六、换一个变量：判断的前提会不会被推翻

先把要维护的那件事拆成两步，这两步平时总被当成一步。

**第一步是判定**：哪些该清、哪些该留、哪些该动。**第二步是执行**：把判定的结果做出来。

资源解释只看第二步——执行要花力气，力气有限，所以得停。而本文要指出的是，绝大多数非停不可的情形，卡在第一步：**判定是在某个时刻做出的，而执行需要时间；在这段时间里，在线活动可能让那个判定不再成立**。

于是变量就出来了：**维护所依据的属性是不是单调的**。

准确说：一项属性对某个系统是单调的，如果它一旦对某个对象成立，在线活动就无法把它逆转回去。

三点澄清，缺一点它就会被读回旧概念。

**第一，单调性是关于判据的，不是关于对象的。** 同一堆东西，按不同的判据去清，单调性可以完全不同。清"已经不可能再被用到的东西"是单调的；清"最近三个月没被用过的东西"不是——它随时可能被用一下，判定当场失效。**换一个判据，一项在线做不了的维护可能就能做了**，这是本文最有实践价值的一句话。

**第二，单调只保证一个方向的安全，因此必然有代价。** 依据过时的判定去执行，会漏掉那些"在这段时间里刚刚变成该清的"东西——它们这一轮收不到，下一轮再收。工程界给这类东西起过名字，叫浮动垃圾。代价是**这一轮清不干净**，而不是**清错**。分不清这两者，是把单调性用错的最常见方式。

**第三，它与资源是正交的。** 一项极耗资源但判据单调的维护，可以在线做，只是慢；一项几乎不耗资源但判据非单调的维护，照样必须停——比如清点仓库里的一百件东西，力气可以忽略不计，但只要收发不停，你就永远得不到一份对的数。

**怎么判——三个可以直接问的问题。**

其一，**逆转追问**：这项维护判定为"该清/该留/该改"的东西，有没有可能因为在线活动而重新变成另一类？答"有"，非单调。

其二，**过时快照追问**：如果我按十分钟前的状态做判定，然后照着做，会做错事还是只会做漏事？只会做漏，单调；会做错，非单调。

其三，**冻结成本追问**：要让这个判定在执行期间保持成立，我需要冻住什么？如果只需冻住一小块（一个分区、一个时点），那就不必停全场——**屏障可以替代停机**；如果要冻住的是全局，那就是停机。

三个问题互相独立，且答案应当一致。若不一致，说明那项维护其实是几件事捆在一起，应当先拆开——这本身就是一条有用的诊断。

## 七、第二根轴：要不要动对象

只有一根轴的东西不是维度，是名字。单调性必须与另一根独立的轴交叉，才能把现实切成可分辨的格子。

第二根轴是：**这项维护要不要移动被维护的对象**。

它与单调性独立，因为"判定会不会失效"与"东西要不要挪地方"是两件事：清点不挪东西但判定会失效；把仓库重新排布是挪东西，而"哪些货该放哪一区"这个判定未必会被收发推翻。

移动之所以单列，是因为它引入了一种与判定无关的失效：**一旦对象被挪走，所有指着它旧位置的东西都错了**。这时你需要的不是一份对的判定，而是一个**所有人都不在使用旧位置**的瞬间——哪怕这个瞬间极短。

两根轴交叉，得到四格。这张表是全文的骨架。

| | **判据单调（判定不会被在线活动逆转）** | **判据非单调（判定会被推翻）** |
|---|---|---|
| **不移动对象** | **格 I · 可全程在线**<br>按过时快照做即可；代价是这一轮漏掉一批，下轮再收 | **格III · 屏障可替代停机**<br>不必停全场，只需在每一次可能推翻判定的动作上截一下 |
| **移动对象** | **格 II · 需要短暂的全局一致点**<br>判定不必重做，但换位置那一刻大家得同时看见同一份地址 | **格IV · 只能停**<br>判定要保鲜、位置要一致，两个条件叠在一起，没有别的走法 |

本文的可裁决排序是：**能被在线化的程度：格 I ＞ 格 II ＞ 格III ＞ 格IV**。 而这个排序与三支给出的都不同：

- 睡眠那一支不按格排序——它把所有离线工作当成同一类，因此预测不出"同一个器官里的两项维护会落在不同格"。

- 内存回收那一支的排序是按**实现难度**：它把格Ⅲ（装屏障）看作比格Ⅱ（短暂同步）更成熟的方案，因为屏障先被发明出来。而本文按**能否在线化**排序，把格Ⅱ排在格Ⅲ之前——理由是：短暂全局一致点的代价是**一次极短的集中停顿**，而屏障的代价是**每一次在线动作的长期加税**。
- 会计那一支只有"能取消/不能取消"两档，四格在它那里塌成两格。

还有一条推论，是这张表最要紧的产物：

> **停机与屏障是同一笔代价的两种支付方式**。停机是集中付——一次长停，代价可见、可测、可被抱怨；屏障是分摊付——每一次在线动作都慢一点点，代价隐形、不可归因、无人抱怨。

第十五章会说明，这条推论有一个相当难看的后果。

## 八、四格的现实标本

**格Ⅰ · 可全程在线**。 数据库里有一类清理工作，专门回收那些已经没有任何事务能看见的旧行。判据是单调的——一个版本一旦对所有当前和未来的事务都不可见，就不会重新变得可见。于是这项清理可以在业务照常跑的时候进行，代价是刚刚变成不可见的那些这一轮收不到，下一轮再收。

同一格里还有：清扫地面、冲刷管道里的沉积、把已经归档的单据从活动区搬走。它们的共同点不是"轻松"，而是**判定不会被在线活动逆转**。

值得注意的是，本文认为脑内液体对代谢废物的冲刷**属于这一格**：废物一旦产生就不会重新变成有用的东西。这条推论有一个可查的后果，第十一章会展开。

**格Ⅱ · 需要短暂的全局一致点**。 同一个数据库里，另一类工作要把数据在物理上重新排布、把索引整个重建。判定本身不难保鲜，麻烦在于位置变了——正在读旧位置的人会拿到错的东西。所以这类工作要么需要一个很短的独占瞬间，要么需要一套"转发"机制：旧位置留一块牌子，写着"东西搬到那边去了"。

现实里的对应物很多：图书馆换架位时那半小时的闭馆；工厂换模；把一个部门整体搬到另一栋楼——**难的从来不是搬，是搬的那一刻谁也别来找他们**。

**格Ⅲ · 屏障可替代停机**。 仓库盘点是最标准的标本。判据"这一区现在有多少"会被收发持续推翻，所以传统做法是关门。而循环盘点做的事情正是屏障：**只冻结正在点的那一区**，其余照常。冻结的不是时间，是那一小块上的推翻动作。

同一格里还有：审计的截止日（在时点上截一刀，之后的算下期）、代码库在大规模改造期间的功能冻结、手术前的禁食与备皮（把"胃是空的"这个判定保住）。

**格Ⅳ · 只能停**。 判定要保鲜、位置要一致，两个条件叠在一起。

道路的面层铺设是干净的标本：新铺的沥青在固化前既不能被压（判定"这一段已经铺好"会被一辆车当场推翻），铺层本身又在改变路面的物理位置与标高。于是没有别的走法，只能封路。

再比如全场停业的物理盘点加重新布局；比如一台机器要拆开来换掉承力件；比如那种必须把整套账停下来、逐笔重新归属的清算。

四格摆出来之后，一件事变得清楚：**我们平时说的"这活儿必须停机"，混了三种完全不同的情况**（Ⅱ、Ⅲ、Ⅳ），而这三种之间的差别，不比它们与格Ⅰ的差别小。

## 九、一条已经被算清楚、却没人搬出来的读数

本文最要紧的那条判据，不需要新实验，它在工程里已经被算了三十年，而且算得比任何思辨都精确。这一章把那三条现成的读数摆出来。

**读数一：安全的方向是不对称的。**

在过时的快照上做判定，会犯哪种错？工程界的答案很明确：**只会漏，不会错**。按快照判为"该留"的，可能其实已经该清了——它这一轮被放过；而按快照判为"该清"的，一定真的该清——因为那条性质不可逆转。

这个不对称正是单调性的全部内容，也是本文与"资源"解释的第一处分岔：**如果停顿是为了资源，那么错误应当是随机的、双向的；而实测的错误是单向的。**

**读数二：代价有名字，而且是"下一轮再收"。**

那些在执行期间刚刚变成该清、却因为快照里还没变而被放过的东西，有一个专门的名字：浮动垃圾。它不被当成缺陷，而被当成**这种做法的定价**——你用"清不干净"换"不用停"。

这条读数给本文提供了一个可迁移的诊断：**凡是一项维护被搬到线上之后，出现了"每次都清不干净、但下一次能补上"的现象，那就是单调性在起作用，而不是效率不够。** 反过来，如果搬到线上之后出现的是**错清**（把该留的清掉了），那说明判据根本不单调，这次在线化是错的，应当立刻退回。

**读数三：停不停是速率的函数，而且会自动切换。**

有一套实现会持续比较两个速度：程序制造新对象的速度，和回收清理的速度。若前者持续快过后者，就逐步削减程序自己的时间配额；极端情况下配额降到零——**系统自动退回了停机**。

这一条把"停机 vs 在线"从架构选择变成了连续量。它同时给出一个反直觉的预测：**在线化做得越成功，被维护的系统越可能加速制造需要维护的东西**——因为在线化把资源还给了它。这条预测在那个领域已经被观察到：有实现报告过，改成并发之后如果不加抑制，内存占用会涨到数倍。第十五章会把这条反噬推广出去。

三条读数合起来，恰好就是本文的机制陈述。而这一支之所以没有把它搬出来，理由在第十七章。

## 十、三支各自为什么到不了这一条

**睡眠那一支到不了，因为"离线"在它那里是一个时段，不是一个条件。**

它的全部工作是问"那段时间里发生了什么"，而不是问"为什么非得那段时间"。于是三派各自给出的不可取消理由——边调边加、回放会被当成现实、通道打不开——被笼统地归成同一句"必须离线"，三种性质完全不同的约束因此从来没有被分开。

这里可以给出一条判决性的差别。睡眠那一支预测：**离线期间的各项工作，其不可取消性应当是同一档的**——它们同属"睡眠功能"。本文预测：**同一个器官里的不同维护应当落在不同格**，因而它们被剥夺时的后果形状不同——非单调的那些一旦被打断就立刻出现可测的功能损害，而单调的那些只表现为"清得慢、积得多"，且可以靠延长时间补回来。第十六章把这条做成检验。

**内存回收那一支到不了，因为它把自己的判据当成了实现细节。**

它确实把条件算清楚了，但它的表述从头到尾绑在自己的对象上：可达性、写屏障、疏散、根集。这些词没有一个是可迁移的。而"可达性一旦丧失便不可恢复"这句话背后那个一般形式——**判定所依据的属性是否可被在线活动逆转**——它从来不需要说出来，因为它只有一个应用场景。

一个领域把某个一般性质当成自己的局部事实，这不是疏忽，是分工的正常代价。

**会计那一支到不了，因为它的两个理由不在同一层。**

"技术进步"解释了关门为什么可以不要，"内控要求"解释了刀为什么必须划。前者是关于能力的，后者是关于规范的，两者推不出对方，也预测不了下一项维护属于哪一类。本文给的是同一层上的一个变量，因此可以预测：**只要一项会计程序所依据的判定是单调的，它迟早会被持续化；只要不是，它就会被保留为一个时点，无论技术多好。**

**最后一句，关于三支共同的默认。**

三支都默认停顿是为了腾出资源，这在各自的场地里都合理：生理学关心能量与代谢，工程关心 CPU 与内存，会计关心人手与工时。资源确实存在，也确实是约束之一。本文所做的，只是把另一个一直被当作背景的东西挪到前台，并且发现——**一旦它成为变量，三对互相取消的立场同时被安顿，而且各自的适用范围恰好对应四个格子中的一个或两个。**

## 十一、它在别处也兑现

一个判断如果只在它出生的领域成立，那是经验规律，不是辨别维度。这一章把它放到别处，每一处都要求它给出一件具体的、可查证的预测。

**其一，脑内的两项维护应当分属两格。** 按本文，代谢废物的冲刷是单调的（废物不会变回有用），因而**原则上可以在线**，只是慢；而"哪些痕迹值得保留"这个判定会被清醒时的新经验持续推翻，因而**必须离线**。预测：剥夺睡眠时，两类后果的形状不同——记忆类的损害立刻出现且**补睡不能完全追回**（那一次的判定窗口过去了），代谢类的指标则表现为**积累量随剥夺时长单调上升，且补睡后可基本追平**。这条差别可以在现成的剥夺实验里查，因为两类指标本来就在同一批研究里被测过。

**其二，数据库的两类维护。** 回收不再可见的旧版本可在线做（单调、不移动）；重建索引与物理重排需要独占或转发（移动）。预测：**同一套系统里，能被做成"并发版本"的那些维护，恰恰是判据单调的那些**，而不是"负载较轻的那些"。这条只要翻一遍任何一个成熟数据库的功能演进史即可判。

**其三，仓库盘点。** 循环盘点之所以能替代停业盘点，是因为它把冻结从全场缩到一区。预测：**收发越频繁的区，循环盘点的差异率越高**，且这个相关强于"该区货值"或"该区面积"与差异率的相关。

**其四，审计截止日。** 它是快照不是停机。预测：**技术进步会持续侵蚀"关门盘点"，但不会侵蚀截止日**；而真正会挑战截止日的，是让"期间归属"变得非单调的会计安排（可撤销的收入确认、可追溯调整的合同条款）——预测**这类安排一出现，围绕截止日的争议与舞弊就集中在它们身上。**

**其五，代码库的功能冻结。** 大规模改造依据的是"当前的调用关系"，而并行开发持续推翻它，所以要冻结；而纯粹的格式化不改变语义，判定单调，可以随时做。预测：**一个团队里，需要冻结期的改造与不需要冻结期的改造，其分界不在改动行数上，而在"改动是否依赖一份全局的当前状态"上。**

**其六，道路养护。** 面层铺设必须封路（非单调＋移动），划线与清扫可以在线（单调、不移动）。预测：**同一条路上各项养护作业的封路时长，与作业本身的工时相关很弱，与"作业期间车辆通过会不会毁掉刚做完的部分"强相关。**

**其七，手术与制动。** 术区固定不是为了省力，是为了保住"已经对好了"这个判定。预测：**同一部位的两类术后医嘱——限制活动与营养支持——其依从性要求的严格程度不同**，前者是判定保鲜，后者是资源供给，而临床上前者的违反后果更陡峭、更不可补救。

**其八，法律与修订。** 一部法在大修期间通常伴随过渡条款，把"依旧法成立的事"锁住。预测：**过渡条款的复杂度，与新旧法之间"某一事实的定性会不会被逆转"强相关**——纯粹增补的修法几乎不需要过渡条款。

**其九，组织的架构调整。** 判定"谁向谁汇报"要重排，同时人和职责在移动，属于格Ⅳ。预测：**架构调整期间的效率损失，与调整涉及的人数相关很弱，与"调整期间业务是否仍在产生需要归属的决定"强相关**——旺季调整比淡季调整昂贵得多，而这不能用人数解释。

**其十，同侪评审与投稿截止。** 会议的截稿时间是快照（单调，属于哪一轮不会被逆转），期刊的滚动收稿则把它取消了。预测：**取消截稿的期刊，其评审争议不会因此上升**；而真正引起争议的，是那些让"属于哪一轮"变得可逆的规则（无限次大修、可撤回重投）。

十处兑现，十个可查的读数。但真正让这条判断长住的，是下一章那两处**反过来迫使它让步**的地方。

## 十二、两处反过来加约束的地方

只会附和的例证是装饰。真正说明一个判断有承重能力的，是它在某处遇到阻力并被迫让步。本文遇到两处。

**第一处：速率会把单调性的好处吃掉。**

按第七章那张表，格 I 的维护可以全程在线。但工程那一支的一条实测把这句话打了个折：如果被维护的系统制造新麻烦的速度持续快过清理的速度，那么无论判据多么单调，在线化都会失败——积压只会越来越大，最后仍然要停下来。有一套实现干脆把这件事做成了自动机制：制造得太快，就削减制造者自己的时间配额，极端情况下削到零，也就是自动退回停机。

这一条逼本文让步两次。

其一，**单调性是"可以在线"的必要条件，不是充分条件**。还要加一个速率条件：在线清理的产能必须能追上在线的生成。第七章那张表因此只回答"能不能"，不回答"够不够"。

其二，更要紧：**在线化本身会抬高生成速率**。因为它把原本被停顿占去的时间还给了在线活动，而在线活动多干的那些事，又会产生新的维护需求。于是存在一个自我加速的回路，而它的存在意味着——**一次成功的在线化，可能在一段时间之后把系统推回到必须停机的状态**，且那时的停机会比原来更长。这条推论我原本不会写出来。

**第二处：屏障不是免费的，而且它的代价长在不该长的地方。**

第七章说停机与屏障是同一笔代价的两种付法。但工程那一支的细节表明，这两种付法在**分布**上完全不同：停机的代价集中在少数人身上的少数时刻（那几秒谁都干不了活），而屏障的代价摊在**每一次在线动作**上——每一笔写操作都要多做一点检查。

这一条逼本文再让步一次，而且这是全文最重要的一次修正：

> **在总量相等的情况下，两种付法并不等价——因为它们被谁看见、被谁归因，是不同的。** 停机是可见的、可被计时的、会被抱怨的；屏障是不可见的、不可归因的、没有人会为它开会。于是一个组织在两者之间做选择时，即使总代价更高，也会系统性地偏向屏障。

于是本文的处方从"判断单调性、能装屏障就装"退到一个更窄也更可做的位置：**装屏障之前，先问这笔分摊的代价由谁承担、有没有人测得到它**。测不到的代价一定会被无限追加，直到某个不相干的地方崩掉——而那时没有人会把它算到这次在线化头上。

这两处让步都不是补丁。它们把本文从"单调就能在线"这个过宽的说法，推到了两个更具体的量上：**生成与清理的速率比**，以及**分摊代价的可见性**。

## 十三、与最接近的几种说法划清界线

一个判断最危险的时刻不是被反驳，是被读者顺手归到某个他早就知道的名目底下。这一章逐个处理，每一处都必须落到"同一个案例，那个说法判 A，本文判非 A"。

**与"资源争用"这一族。** 这是最容易吞掉本文的一位：停机是因为维护和在线活动抢同一副硬件。分离线在两处：其一，本文预测**一项几乎不耗资源的维护照样可能必须停机**（清点一百件货，力气忽略不计），资源解释在这里预测无需停机；其二，本文预测**加倍投入不会缩短非单调维护所需的冻结**——你可以多派十个人去点货，但只要收发不停，你仍然得不到一份对的数。资源解释预测人多就快。

**与"批处理与流式"的区分。** 计算里有一组老区分：攒一批再算，还是来一条算一条。它与本文共享很多例子，但自变量不同：那一组的自变量是**吞吐与延迟的取舍**，本文的是**判定会不会失效**。可裁决之处：存在判据单调却仍适合批处理的情形（纯粹为了吞吐），也存在判据非单调却被迫做成流式的情形（此时必须配屏障）——那一组不预测后者需要屏障。

**与"事务与隔离级别"。** 数据库里有一整套关于"一个操作看到的世界是否一致"的成熟理论，快照隔离几乎就是本文说的"过时快照"。分离线：那一套解决的是**读的一致性**（我看到的是不是一份自洽的世界），本文问的是**执行的有效性**（我按看到的去做，做完还对不对）。二者可以分开：一个隔离级别完美的系统，照样可能在维护上撞见非单调——因为它的"对"是对读者说的，不是对改造者说的。

**与"临界区与锁"。** 锁是屏障的一种实现，但锁的理论关心的是互斥与死锁，不关心**被保护的判定是什么性质**。可裁决之处：本文预测**判据单调的维护不需要锁也能正确**（只会漏不会错），而按锁的直觉，任何并发访问都该加锁——两者在这一格给出相反的建议，而且这是工程上可以直接验的。

**与"维护窗口"这一实践概念。** 它是现象的名字，不是解释：它告诉你这段时间要停，不告诉你为什么这段必须停、那段不必。本文提供的正是它缺的那个判据。可裁决之处：本文预测，**在维护窗口里做的各项工作，其可迁移到线上的顺序，与它们的判据单调性一致，而与它们的时长无关**——而实践中安排窗口靠的恰恰是时长。

**与睡眠功能的三派。** 分离线已在第十章给出：本文预测同一个器官里的不同维护分属不同格，因而被剥夺时的后果形状不同（陡峭且难补 vs 单调累积且可补），三派都不作这个区分。

最后指认最近的那一位：**资源争用**。本文之所以仍不能被它吸收，只靠一件事——它的自变量是"够不够用"，本文的是"还成不成立"，而这两者可以取到相反的值。那个"耗资源极少却必须停"的情形是否真的普遍存在，是本文最该被攻击的地方。

## 十四、它不适用在哪

**其一，纯粹的物理不可能。** 有些停顿与判定毫无关系：混凝土要养护七天，胶要固化，药要起效，伤口要长。这些地方停的是"必须等"，不是"必须冻"。把本文推到这一类上，会得出"只要判据单调就可以在线"的荒唐结论。粗略的分辨办法：问一句"如果我在这期间碰它一下会怎样"——答"结果会错"是本文的地盘，答"什么也不会变，只是还没好"不是。

**其二，安全性质与活性性质混在一起的场合。** 本文全篇谈的是"做得对不对"。但很多停机的真正理由是"出事了谁负责"——即使技术上可以在线，规程也要求停，因为停下来才有人明确地在场、明确地签字。这一类停顿由责任结构决定，不由单调性决定，本文对它没有说法。

**其三，单调性本身可被制度改变的场合。** 第六章说"换一个判据可能就能在线"，这在技术系统里是设计自由，在人的系统里往往不是——判据常常写在法规、合同、职业规范里，改它的成本远高于停一次机。本文能指出"换判据可以省掉停机"，但不承担"该不该换"这个问题。

**其四，我说不清"一小块"到底多小。** 第七章说屏障是"只冻住可能推翻判定的那些动作"。可在很多现实系统里，判定所依赖的东西盘根错节，冻住"相关的那部分"最后等于冻住全部。哪些系统能被局部冻结、哪些不能，本文给不出判据，只能说它与系统的耦合程度有关——而这句话近乎同义反复。

**其五，极短与极长的两端。** 本文的读数以"一次维护周期"为单位。对毫秒级的调度与对以年计的大修，它都很可能失效：前者的开销全在切换本身，后者的主要约束是预算与政治。

把五条合起来，本文实际主张的范围要比标题窄得多：**在一项维护有明确判定步骤、判定与执行之间存在时间差、且系统在这段时间里仍可能被在线活动改变的场合，判据的单调性决定它能否被在线化，而移动与否决定它需不需要一个全局一致的瞬间。** 超出这个范围的部分，我不认账。

## 十五、按它设计干预，会怎样出事

**第一种：屏障的隐形税会被无限追加。**

上一章已经说了机制，这里说后果。一旦一个组织学会"用屏障换停机"，它会反复用——因为每一次都显得像进步：那个讨厌的停机窗口没有了，而多出来的成本没有一个部门认领。三五次之后，在线的每一个动作上都挂着一串检查，系统整体变慢，而没有人能说清是哪一次改造造成的。

对治办法只有一个，而且不彻底：**每装一道屏障，同时给出它的分摊代价的测量方式，并且事先约定这个数由谁盯着。**测不到的代价一定会被继续追加。

**第二种：在线化把资源还给在线活动，于是麻烦长得更快。**

第十二章那条自我加速的回路，在实践里的样子是：把维护搬到线上之后，业务侧发现"现在可以多做一点了"，于是多做；多做产生更多需要维护的东西；一段时间之后积压反而更大，最终不得不停一次更长的。

而这次更长的停机**不会被归因到当初那次在线化**——它会被归因为"业务增长太快"。

**第三种：把判据改成单调的，可能是在悄悄改掉目的。**

第六章那句最有实践价值的话有一个阴暗面。"清最近三个月没用过的东西"改成"清已经不可能再被用到的东西"，确实把非单调改成了单调，于是可以在线做了——但这两条判据清掉的东西完全不同，后者要保守得多。**在线化的收益是真的，而它是用"清得更少"换来的**，且这个变化藏在判据的措辞里，不会出现在任何一份改造总结里。

由此得到一条硬规矩：**凡是为了在线化而改判据的，必须同时报告改前改后清理量的差。**报不出这个差，就是把目的换掉了还当成效率提升。

**第四种：把"能在线"当成"应该在线"。**

有些停顿承担着与维护无关的功能：停业盘点那一天，全店的人都在场，很多平时没人管的问题会在那天被顺手发现；睡眠期间发生的事情里，也许有一些至今没被任何一派测出来。**取消一个停顿，等于同时取消掉所有搭在它上面的东西**，而搭在上面的那些通常没有名字，也就不会出现在收益核算里。

**第五种：本文自己会被优化。** 留到第十八章。

## 十六、怎样算它错

这一节是全文最要紧的部分。前面的论证如果没有它，就只是一个说得通的故事。

**其一，最便宜、现成数据就能做的一条。** 取任意一个成熟数据库或运行时的功能演进史，把历年从"需要独占"变成"可并发"的维护列出来，逐项标注两件事：判据是否单调、是否移动对象。

- 本文预测：**成功在线化的那些，压倒性地集中在判据单调的一格**；而始终未能在线化的那些，集中在"非单调且移动"。
- 资源解释预测：成功在线化的应当集中在**开销小**的那些。
- 若在线化的成败与开销的相关强于与单调性的相关，本文的核心变量选错了。

这条不需要被试、不需要经费、不需要任何机构配合。

**其二，两类剥夺的后果形状（打睡眠那一支）。** 按本文，记忆类维护非单调、代谢类维护单调。预测：剥夺之后，记忆类损害**陡峭出现且补睡难以完全追回**，代谢类指标**随剥夺时长单调累积且补睡后基本追平**。睡眠那一支预测两类同属"睡眠功能"，受损与恢复的模式应当相似。这两组数据在现有文献里本来就分别存在，需要的只是把它们放在同一张图上比。

**其三，耗资源极少却必须停（打资源解释）。** 设计一个极端案例：一项判定复杂但执行几乎零成本的维护（如只需在一份清单上打勾），在收发持续的条件下运行。本文预测**仍然需要冻结**；资源解释预测无需。这条可以在一个真实仓库里用一天做完。

**其四，正向读数。** 本文为真时应当出现一件可测的事：**同一系统里各项维护所需的冻结时长，与"该项判定在单位时间内被在线活动推翻的次数"正相关，而与该项维护本身的工时相关很弱**。这是正向的——它要求本文预言的东西出现，而不只是要求反例不出现。

**其五，会让本文尴尬的一条。** 第十二章说在线化会抬高生成速率，从而可能把系统推回停机。若追踪若干成功在线化的系统若干年，发现停机需求持续下降而非回升，说明那条自我加速的回路要么不存在、要么被别的机制压住了，第十二章第一处让步应当撤回。

**其六，两个明写交出去的洞。**

第一个洞：我给不出"局部冻结可行性"的判据（第十四章第四条）。哪些系统能被局部冻住、哪些一冻就冻住全部，本文只能说"看耦合程度"，而这句话接近同义反复。

第二个洞：我说不清**判据的单调性本身能不能被判定**。第六章给了三个问题，但这三个问题的答案往往依赖对系统的完整了解——而如果你已经完整了解它，很多维护问题本来就不难。这个循环我没有解开。

## 十七、为什么这件事一直没被看见

一个变量如果这么要紧，为什么三支都漏了？把这个问题答清楚，本文才算完整——否则最可能的解释是：它没那么要紧。

答案有三层，三层都不是疏忽。

**第一层：在最常见的场合里，两个原因永远同时出现。**

停机的时候，资源确实被腾出来了，判定的前提也确实被冻住了。两件事同时发生，而且都能解释为什么要停。要把它们分开，你需要一个反常的案例——**耗资源极少却必须停**（清点一百件货），或者**极耗资源却可以在线**（大规模的沉积冲刷）。这两类案例在日常里不是没有，而是没有人有理由把它们摆在一起看，因为它们分属完全不同的行当。

**第二层：这条判据在唯一算清楚它的地方，被写成了局部事实。**

工程那一支把条件算得极精确，但它的每一个词都绑在自己的对象上。一个领域把某个一般性质表述成自己的专门术语，是分工的常态；而术语一旦成形，就同时起了两个作用——**它让这条性质在本领域内可用，也让它在本领域外不可见**。别的行当读到那些词，会正确地判断"这是计算机的事"，然后合上。

**第三层，也是最要紧的：停机留下记录，而判定失效不留记录**。

停机是有痕迹的：它有开始时间、结束时间，会被写进日志、排进日程、算进损失。而"如果我们不停，判定会失效"这件事，**在停了的情况下永远不会发生**，因此不产生任何数据。它是一个反事实，而反事实不会自己留下痕迹。

于是出现了一个稳定的偏差：**所有关于停机的讨论，都建立在留痕的那一侧**——多久、多贵、能不能缩短。而不留痕的那一侧——为什么非停不可——被默认成"因为要干活"，因为这是唯一一个能被日志证实的解释。

这一层还给出一条可查的推论：**在那些偶尔真的试过"不停"的地方，这件事应当早就被知道**。确实如此——盘点时没有封存出入库的仓库、大修期间没有冻结开发的团队、边铺边通车的路段，事后都会得到一批说不清来路的错误，而当事人往往把它归为"配合不好"或"人为疏忽"。**他们撞见的正是判定失效，只是那次失效没有名字，于是被算到人头上**。

## 十八、本文自己也是标本

如果这条判断成立，它首先应当适用于写它的这件事。

**第一件要认的账：这篇文章依据的判定，正在被在线活动推翻**。

它的三支源头里，有两支在持续更新——睡眠功能那一支每年都有新的实验，工程那一支每一代运行时都在挪动"必须停"的边界。也就是说，本文所依据的"哪些工作能在线、哪些不能"，在我写它的这段时间里就已经在变。

按本文自己的表，这属于**判据非单调**。而我写它的方式是什么？是在一份**过时的快照**上做判定——读到某个时点为止的材料，然后照着写完。按本文的判据，这么做的结果应当是"只会漏，不会错"——可那条保证只在单调时成立。**在非单调的对象上用快照，会错**。

我没有办法在这篇文章内部消解这一点。能做的只有两件：把它写在这里；以及在第十六章把那条最便宜的检验设计成**任何人拿现成资料就能重跑**的形式——重跑一次，就等于把快照刷新一次。

**第二件要认的账：这篇文章本身是一次没有冻结期的改造**。

它要动的是三支各自的地基（睡眠功能的归属、工程判据的适用范围、会计程序的分界），而这三支在此期间照常运行、照常发论文。按本文第十一章第五条的说法，这类改造需要功能冻结——而学术里没有这种东西，也不该有。

于是本文只能落在最糟的那一格：**判定会被推翻，而且它要动的东西正在移动**。这不是修辞上的自谦，它有一个具体后果——本文很可能在某个细节上已经过时，而我不知道是哪一个。

**第三件，也是最不舒服的一件**。

本文批评三支把停顿的理由默认成资源，而本文自己也有一个被默认的东西：**它默认"维护"和"在线活动"是两件可以分开的事**。在机器和仓库里，这个划分很清楚；在人和组织里，它常常不成立——一次谈话既是在做事也是在修复关系，一堂课既是在教也是在校准教的人自己。对这一类"维护与运行不可分"的场合，本文的整张表都不适用，而我在正文里只用一句话带过。

我不打算假装这里有什么深意。我只是和我所描述的那三支一样，把此刻对我最方便的那个划分，当成了不需要说明的背景。

## 十九、结论，与一条写死日期的赌注

先把结论收拢成三句。

第一句：**关于"为什么非得停下来"，三支成熟传统给出的立场互相取消**——一支说离线是不可取消的根基，一支的整个纲领是消灭停顿，一支实践上取消了关门却坚决保留划刀。三支都不是错的，但三支都把停顿的理由默认成了同一件事：腾出资源。

第二句：**真正决定能不能不停的，是判定所依据的属性是不是单调的**——一旦成立，会不会被在线活动逆转。单调就可以在过时的快照上做，代价是这一轮漏掉一批、下一轮再收；不单调就要么停下来，要么在每一次可能推翻它的动作上截一下。再加上第二根轴（要不要移动对象），四种情形各有各的走法。**停机是为了冻结判断的前提，不是为了腾出资源**。

第三句：**因此"能不能不停机"是个问错的问题**。 该问的是两句：我们依据的那个判定是不是单调的；如果不是，能不能只冻住会推翻它的那一小块。而选择屏障之前还有一句必须先问——**这笔分摊的代价由谁承担、有没有人测得到它**。测不到的代价一定会被无限追加，直到某个不相干的地方崩掉，而那时没有人会把它算回这次改造头上。

剩下的事，是让它可以被杀死。

**赌注：到 2032 年 12 月 31 日**。

条件一：在此之前，第十六章第一项那份清单已被做出来——任取一个成熟运行时或数据库，把历年从"必须独占"变成"可并发"的维护逐项标注判据单调性、是否移动对象、以及开销量级。

条件二：结果可从公开渠道取得。

**若两个条件成立，而在线化的成败与开销量级的相关强于与判据单调性的相关——本文作废**。 不是需要修正，是作废：因为第六章那个变量是全文的承重，它若不如"开销"能解释数据，剩下的都只是把三支的话换了个说法。

再加一条更快到期的：**到 2028 年 12 月 31 日**，第十六章第三项那个仓库实验——一项判定复杂而执行几乎零成本的维护，在收发持续的条件下是否仍然必须冻结——应当已经有人做过。它需要一间仓库、一天时间、一份清单，不需要任何新技术。

如果三年之内连这个都没有人做，那不是本文的错，但也不是本文的功劳。它只说明这条判断还没有进入任何人真正会去检验的问题清单——而按本文自己的机制，这是可以解释的：**它要检验的那件事，恰恰是一个只有在"不停"的时候才会显形、而所有人都习惯了先停下来的东西**。

## 附：English abstract

### Why It Has To Stop: maintenance goes offline not to free resources but to freeze the premises of a judgement

**Abstract.** Some maintenance requires the maintained system to halt. The default explanation is resource contention, yet it fails on two counts: some halts cannot be bought away with more resources, and others vanish after a modest technical change. Three unrelated traditions take mutually cancelling positions — sleep research treats the offline period as irreducible, concurrent

garbage collection has spent three decades abolishing pauses, and accounting has abolished the closed-store count while retaining the cut-off date. This paper argues that all three treat as background what should be the variable: **whether the property a maintenance judgement relies on is monotone** — once true, not reversible by online activity. Monotone judgements can be executed against a stale snapshot, at the cost of floating garbage (missed this round, caught the next); non-monotone ones require either a halt or a barrier intercepting every action that could invalidate them. Crossing monotonicity with object relocation yields a 2×2 and the claim that **halting and barriers are two payment schedules for the same cost** — one concentrated and visible, the other amortised and unattributable. Ten cross-domain redemptions are given; two constraints force concessions. Six falsification conditions follow, the cheapest requiring only a public changelog, plus a dated bet.

**Keywords** monotonicity; halt versus barrier; stale snapshot; floating garbage; online maintenance; visibility of amortised cost

## 参考文献

1. Dijkstra, E. W., Lamport, L., Martin, A. J., Scholten, C. S., & Steffens, E. F. M. (1978). On-the-fly garbage collection: an exercise in cooperation. *Communications of the ACM*, 21(11), 966–975.

2. Yuasa, T. (1990). Real-time garbage collection on general-purpose machines. *Journal of Systems and Software*, 11(3), 181–198.（快照式标记与写屏障）

3. Jones, R., Hosking, A., & Moss, E. (2011). *The Garbage Collection Handbook: The Art of Automatic Memory Management*. Chapman & Hall/CRC.（浮动垃圾、疏散与安全点的标准处理）

4. Detlefs, D., Flood, C., Heller, S., & Printezis, T. (2004). Garbage-First garbage collection. *Proceedings of ISMM '04*.

5. Tononi, G., & Cirelli, C. (2014). Sleep and the price of plasticity: from synaptic and cellular homeostasis to memory consolidation and integration. *Neuron*, 81(1), 12–34.

6. Wilson, M. A., & McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172), 676–679.

7. Rasch, B., & Born, J. (2013). About sleep's role in memory. *Physiological Reviews*, 93(2), 681–766.

8. Xie, L., Kang, H., Xu, Q., et al. (2013). Sleep drives metabolite clearance from the adult brain. *Science*, 342(6156), 373–377.

9. Committee of Sponsoring Organizations of the Treadway Commission (2013). *Internal Control — Integrated Framework*.（循环盘点与持续监控的制度背景）

10. International Auditing and Assurance Standards Board. *ISA 501: Audit Evidence — Specific Considerations for Selected Items*（存货监盘）；*ISA 330*（截止测试的性质）。

11. Bernstein, P. A., Hadzilacos, V., & Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.（快照与隔离级别）

12. PostgreSQL Global Development Group. *Routine Vacuuming*；*REINDEX CONCURRENTLY*（在线与独占两类维护的官方划分）。

**文献说明**：本文对上述文献均为义释，不使用直接引文；卷期页码须按所用版本核校。第三、九两章所用的工程读数（单调性使陈旧快照上的遍历安全、浮动垃圾作为其定价、以及分配速率过快时自动退回停机）取自第 2–4、11–12 条一族的标准处理；**把它们读成"一切维护的一般判据"是本文的推论，不是这些文献的主张**，正文已注明。睡眠三派的功能归属之争，本文只取其共享的方法学一步，不对功能归属本身站队。本文的核心检验**均未执行**，交出的是命题、判据与设计。

*字数：正文约 1.7 万字 · 版本 v1.0 · 2026-08-01*

*人机分工声明：本篇由 Claude 依王德生的方法体系执行三家源材料的选取、冲突判定、占位检查、变量替换与全部行文；文中实证读数均引自公开文献，未做新的数据采集。*

---

## 附：与本栏另外两篇的关系

本栏另有三篇与本篇同形：《可预期的尺子》说判据能不能被提前推断，决定了一个领域还提得出什么问题；《谁说了这一次到此为止》说「一次」的边界由谁划，决定了什么被当作可比较的两次；《说过去就没了》说信道让不让你回头，决定了哪些结构长得出来。四篇合起来是同一个形状的四处——**决定长出什么的，往往不是内容，而是内容所在那条管子的性质**；本篇问的则是：那条管子什么时候非得先停下来不可。

这一篇撞的三家，与可核对的出处
**本篇撞的是站外三家传统，不是站内学员论文**（同本栏之八、之九、之十的口径）。三家都在处理同一个问题——有些维护为什么必须让被维护的东西先停下来——而给出的立场互相取消：一家说离线是不可取消的根基，一家的整个研究纲领是消灭停顿，一家在实践上取消了关门却坚决保留划刀。其中第二家已经把条件算到了极精确的程度，只是把答案写成了自己领域的实现细节。正文第二至第四章逐一交代三家说到了哪一步，第五章把矛盾写死，第九章摆出三条已经被算清楚、却没人搬出来的读数，第十三章再与另外五种最容易被混为一谈的说法逐条分界——**其中第一位是本文承认最可能吸收掉自己的那一个**。

系统神经科学 · 睡眠功能之争
整晚在做什么：普遍下调、选择性重演，还是代谢废物的冲刷
三派对功能归属互不相容，却在一件事上高度一致：这些工作必须离线做，而且离线不能被取消。**与本文的分离线**：三派给出的不可取消理由其实是三种（边调边加、回放会被当成现实、通道打不开），被笼统记成「必须离线」之后，那条决定哪一项**有可能**被搬上线的区分就永远得不到。
计算机系统 · 并发内存回收
把必须暂停的窗口从几秒压到亚毫秒：快照式标记、写屏障、疏散与安全点
它为了取消停顿，被迫把停顿的理由拆开了：正因为「一个对象一旦不可达就永远不可达」，才可以在过时的快照上遍历；代价有名字，叫浮动垃圾。**与本文的分离线**：它把这条判据写成了自己的局部事实（可达性、写屏障、疏散），而这些词没有一个是可迁移的——本文要做的正是把它搬出来。
会计与内控 · 盘点与截止
关门盘点被循环盘点与持续审计取代，而截止日不可取消
同一个行当取消掉了关门，却取消不掉划刀，而它给的两个理由（技术进步、内控要求）不在同一层。**与本文的分离线**：本文预测，只要一项程序所依据的判定是单调的，它迟早会被持续化；只要不是，它就会被保留为一个时点，无论技术多好。