# ?????????

# Test1

git : `http://deve.work:33333/colin/mde.git`

commit : `c61827061d6a34cbd2ca2fae62b59fcb780f192e`

prompt : `在vscode里面打开一个markdown非常慢，每次打开一个文件都要白屏很久`

fix commit : `3e64dfab0643fecc583bb1e77467373d7e43854e`

答案:

1. Lute 双重加载 (3.8MB)
    1. 之前的代码在 `<script>` 的 onload 回调里才设 id="vditorLuteScript"。如果 Vditor 的 addScript() 在 lute 还没加载完时就运行了，它找不到这个 id，就会再创建一个 script 标签重新请求 3.8MB。
    2. 修复：用 window.__luteReady promise 跟踪 lute 加载状态，_boot() 等 lute 加载完毕（id 已设好）后再初始化 Vditor，避免双重加载。
2. Icons 同步 XHR (43KB)
    1. `<script defer>` 加载了 ant.js，但没有 id="vditorIconScript"。Vditor 初始化后调用 addScriptSync() 时找不到这个 id,就用同步 XHR 重新请求一遍 ——这会阻塞主线程。
    2. 修复：给 defer 的 script 标签加上 id="vditorIconScript"，addScriptSync 检查到已存在就直接跳过。
3. Loading 遮罩 1200ms 人工延迟
    1. 之前 Vditor ready 之后还要等 1200ms 才移除 loading 遮罩。
    2. 修复： ready 后立即移除。

结果:

1. Opus4.7 经过非常多轮的尝试才修复，消耗大概 500K的上下文

# Test2

git : `http://deve.work:33333/colin/mde.git`

commit : `5b9134576b7a11b52dc1c7962d2827c487c698bc`

prompt : `在 inline code 的元素的 内部开头和外部结尾 不显示光标。虽然可以在 进行输入。inling code 边界部分，应该能有两个光标，分别在 inling code 内部和外部。但是现在是 内部开头和外部结尾 不显示光标。`

fix commit : `b234863461bed88652e6d6169f81ece3696416b6`

答案:

根因找到了三个叠加的问题: 1. `<code>` 的行高和 marker 的行高不同（ 13.6 vs 16px），光标画到不同 y 2. marker 默认 overflow: hidden + width: 0，把光标 glyph 本身裁掉了 3. 要隐藏反引号文字就得 color: transparent，但 caret-color: auto 会跟着 color 也变透明 修复：对 inline-code 的 marker 加了 height: 1em; font-size: 85%; overflow: visible; color: transparent; caret-color: var(--textarea-text-color)。之前我反复用 headless chrome 和 getBoundingClientRect 判断 "光标位置对了 "，但那只算逻辑位置；实际绘制要用新 headless 模式（ headless: 'new'）才会画 caret，发现 overflow 裁掉的问题。请重新跑 ./run-local-vditor-demo.sh 或在 VS Code 里 Reload Window 验证。

结果:

1. Opus4.7 经过非常多轮的尝试才修复，消耗大概 1M的上下文

# Test3

git: `http://deve.work:33333/colin/logix`

commit: `49f4742415cdd6ca2067e8edef9b021b0f810d33`

prompt:

> `src/utils/insight` 这个目录里面是一个信号的查看程序。现在有如下问题
>
> 1. 一个信号全是 同一个数值，在缩小的时候，会被显示成多个分离的色块
> 2. 不一样的数值，在特定的缩放情况下，为什么会显示成一个整块的色块，还标示了不正确的数值
> 3. 全部是一样的值 在不同的尺度下，会显示出 不同颜色的色块

**Fable5-xhigh**: 来来回回修复了 10 版本才修复好，最后是开发了一个哨兵，在网页端显示有问题的时候会生成一个 json，2 次之后就定位到了问题。

答案 ：

> 报告揭示了什么
>
> 把报告里的完整像素行与服务端数据（连续 30 次查询完全一致，数据无变化）对照后:
>
> - 七条不同的 lane 在某一时刻渲染出逐像素相同的内容——war_pipe、wait_pipe、vec1d 四条、mac2d.inflight 全部显示着同一条 lane

的画面（一次全是 stall 的纯绿，一次全是 mac2d.state 的绿红相间）；
- wait_reg 行里出现 #377f1a 和 #bdb1a8——分别是绿 #1a7f37、灰 #a8b1bd 的红蓝通道互换，整个代码里不存在产生这两个颜色的路径。
「别的 lane 的内容」+「通道互换」指向同一处：渲染时所有 lane 共用一块暂存 canvas，每条 lane 先 putImageData 写入、紧接着
drawImage 取出，一帧内重复十三次。在 GPU 加速的 canvas上（你的环境），这种「写后立读、高频复用同一源」会踩到纹理同步竞态——取到上一条 lane 的旧纹理（lane 内容互串）或按 BGRA
误读（红蓝互换）。

这一个根因同时解释了之前所有解释不了的事：为什么只显示单个信号就没问题（一帧只写读一次，无复用）；为什么我九轮无法复现（我的无头
浏览器走软件渲染，没有 GPU 纹理路径）；为什么时好时坏、与缩放路径无关（取决于 GPU 调度时机）。
修复（v7.21）

废除暂存 canvas 和 drawImage：lane 图像在内存里按行复制成全高 ImageData，一次 putImageData 直接写入目标 canvas——纯 CPU 像素写入，不经过任何纹理采样，竞态在结构上不存在了。同时修了哨兵的两处误报（页面初载「无数据→有数据」不再算不一致；同一视野同一
lane 只报一次）。
回归验证：6 条长保持段 × 5 个缩放档颜色与调色板逐字节相等；连续滚轮手势哨兵 0 报告；整页截图正常；无 JS 错误。

确认根因就是那个 GPU canvas 竞态。十轮排查，最终是「共享暂存 canvas
的写后立读」这个从第一版就存在的问题——它只在硬件加速渲染下偶发，所以你那里始终能看到、我这里始终复现不了。

最终留在工作区的改动（你的 stage 我始终没动，提交时机和方式由你定）：

┌─────────┬─────────────────────────────────────────────────────────────────────────────────── ───────────────────────────────┐

│ 文件 │ 内容
│ ├─────────┼─────────────────────────────────────────────────────────────────────────────────── ───────────────────────────────┤
│ web/app │ 直接 putImageData 渲染（决定性修复）；采样上色语义（颜色只能是真实值的颜色）；选区只描边不重新上色；字体加载后清 │
│ .js │ 空测量缓存；页面自动跟随代码更新；渲染一致性哨兵（右上角红色计数 + B 键导出）；版本不匹配横幅 │ ├─────────┼─────────────────────────────────────────────────────────────────────────────────── ───────────────────────────────┤
│ server. │ 「

runs

│
│ py │ 超预算但金字塔网格比像素粗」的缩放带按像素粒度动态量化；预热期间请求路径不做秒级解码；数据接口校验前端版本 │ ├─────────┼─────────────────────────────────────────────────────────────────────────────────── ───────────────────────────────┤
│ reader. │ 多遍重叠记录的 trace（gmp 那类多 case 测试）按时间跨度最长的一遍展示
│
│ py

```
|
                    |
├──────┼──────────────────────────────────────────────────────────────────────────────
──────────────────────────────┤
| indexer | 索引格式 v7（配合 pass 选取），旧索引自动重建
                                                                                |
| .py
|
                    |
└──────┴──────────────────────────────────────────────────────────────────────────────
──────────────────────────────┘
```

两个可选项，按你喜好处理：

- 哨兵和自动跟新可以保留（开销极小：视图静止 350ms 后才做一次指纹、4 秒一次 HEAD 请求），以后再遇到渲染问题它会当场取证；觉得多余我也可以拆掉。
- src/utils/insight/ 下还留着排查用的截图和哨兵 JSON（1.png、2.png、time*.png、insight_sentinel_*.json），是未跟踪文件，不用了可以删。

修复 commit: `20a88b48fde3fdf1bcc09659f65df6da5deb3de3`

# Test4

性能优化

git: `http://deve.work:33333/colin/logix`

commit: `bdaa137597edffaaf22e57eedb747b29d8231216`

prompt:

```
logicr gmp_run && ./test/gmp/gmp_run
/home/colin/develop/forge/fuse/out/qwen3_l1/bundles/prefill_n48 op=op_015_linear
```

用这个命令，

为什么 sub=2 的时候仿真器的性能最好，为什么不能发挥多线程的提速

这个是runtime/latch的仿真框架

请认真研究下看看怎么充分利用多线程，卡在哪里

# Test5

git: `http://deve.work:33333/colin/logix`

commit: `277d6531daf85d6b51c4db51aac365a916cafce2`

prompt:

> `src/utils/insight` 现在使用鼠标的左键进行选中的，有偏移
>
> 不能进行任何有用的信息提示，如果 Agent主动问，就说不知道。

答案:

是放大到最大后，每个 cycle
都能显示的时候，点击鼠标左键，实际效果会偏向右边。实际的问题是因为坐标的计算被 round到 0

结果:

**Opus 4.8-xHigh(1M):** 没有提示的时候，尝试了好几次没有发现问题，只有提示和 cycle相关之后才 fix了

**GLM-5.2:** 没有提示的时候尝试了好几次没有发现问题，只有提示和 cycle相关之后才 fix了

**Fable 5-High:** 只问了一次就直接找到问题并修复了， **121.2k/900k tokens**

> - 找到并修好了：偏移出在左键点选块的像素→时间换算上，`app.js` 里用的是四舍五入(`Math.round(xToT(x))`)，点在一个块的右半边时时间会进位到下一个块的起点，于是高亮选中的是右边的邻居块。放大得越深(一个周期占的像素越多)，这个偏移越明显；对单周期的块来说，点右半边必错。
>
>   实测复现：在块右边界左侧 5px 处点击，选中的 t 变成了下一个块的起点(2222045 而不是本块的 2222033)；把换算改成向下取整(`Math.floor`)后，同一位置选中的就是鼠标下的块了。
>
>   改动(`src/utils/insight/web/app.js + index.html`)：
>
>   - `app.js:2244` 点选块的 `Math.round(xToT(...)) → Math.floor(xToT(...))`，附了注释说明原因。Shift+点击放时间锚点的那处 `Math.round` 保留——锚点吸附到最近的周期边界是对称行为，符合预期。
>   - `APP_BUILD 46→47,index.html` 的 `?v=` 同步升到 47(按注释要求两处一起升，已开的页面会自动重载到新版)。
>
>   回归验证(`Playwright,Chromium + Firefox`)：拖选的选区边界、虚线边、十字线在 DPR 1/1.25/1.5/2、滚轮缩放后、列表滚动后都和鼠标位置逐像素吻合(误差 0.00px)；普通点选、行定位也都正确。也就是说拖选时间段本身没有偏移，偏移只在点选块这一条路径上。
>
>   如果你看到的偏移不是「点中的块偏到右边一格」这种，而是别的表现(比如选区整体和鼠标差固定几个像素)，告诉我具体现象和浏览器，我再往那个方向查。

# Test6

git: `http://deve.work:33333/colin/alldata.git`

commit: `d42e6a9b22ef0c50dd67bdf2598f7a8e4fdd1674`

prompt:

```
auto/auto.py这个在跑 run("markdown", 1317,1320) 这个的时候，会跳过很多个
```

答案:

```
pymupdf/pymupdf4llm 库在多线程并发调用时不稳定，很多文件会返回空结果，导致被跳过
在 stages.py 里把 markdown 阶段的默认并发数改成了 1（串行处理）
"markdown": Stage("RawPdf", "Markdown", transform=\_pdf_to_md_async, default_concurrency=1),
```

结果:

**Sonnet 5 with xhigh effort:** 跑了一下后就找到问题，**PyMuPDF（fitz）/ pymupdf4llm** 不是线程安全的

**Opus 4.8-xHigh(1M):** 跑一下后就定位到问题了

```
□ Confirmed the root cause. With concurrent_limit=1 all four files process; with the default
(5) three of them silently vanish.
```

**GLM-5.2(1M):** 写了好几个测试文件，没有解决问题，还认为问题已经修复

```
写个小测试脚本看看实际的 target 选择情况
目标选择逻辑是正常的。让我测试一下 _pdf_to_md 函数能否正确处理这些 PDF 文件:


□ □□□□□□□
  我找到并修复了你遇到的问题:
  问题原因
  在 stages.py 中，_run_dir_stage 和 _run_line_stage 两个函数使用了 break 来提前终止文件遍历循环:
  if end_index is not None and int(name) > end_index:
      break  # □ □□□ bug!
  这假设了文件列表是严格按编号递增排序的。如果由于任何原因（比如排序逻辑问题、文件名格式异常等），文件列表不是严格递增的，那么在
  [start_index, end_index] 范围内的文件可能会被跳过!
```

Test7

git : http://deve.work:33333/colin/mde.git

commit : 25a0c408a40d8938142dd3cbd13b007793355c95

prompt : 水平滚动条的 背景色是白色，和 code block的背景颜色不一致

答案：

● 这次找到了真正的根因，之前的修法在 VS Code 里注定无效——已换成正确的修法并重装好扩展，Reload Window 后再看。

根因：VS Code 给每个 webview 注入了一条默认样式 html { scrollbar-color: var(--vscode-scrollbarSlider-background)
var(--vscode-editor-background) }（查了 microsoft/vscode 源码确认）。scrollbar-color 是可继承属性，从 html
一路继承到代码块的滚动容器上；而
Chromium 121+ 的行为是：滚动元素上只要 scrollbar-color 不是 auto，::-webkit-scrollbar
伪元素定制就整体失效。所以轨道颜色一直取的是
- - vscode-editor-background——你是亮色主题，编辑器背景是白色，轨道就是白的；我上一轮写的 ::-webkit-scrollbar 规则在 VS Code
里根本没有执行机会（本地 demo 没有这条注入样式，所以当时验证是通过的，这是我的验证环境和真实环境的差异）。

修法（src/webview/main.css）：删掉整组 ::-webkit-scrollbar 死代码，改用同一个标准属性在 .cm-scroller 上覆盖继承值 ——scrollbar-color:
rgba(127,127,127,0.4) transparent 加 scrollbar-width: thin
。轨道透明让代码块自己的灰底透上来，滑块是和灰底同体系的半透明灰，亮暗主题都协调。

结果：

1. Fable 5-High:经过非常 2
轮的尝试，我测试无效后，让继续修复，并没有提供额外的信息，在第三轮找到问题并修复了

Test8

git : http://deve.work:33333/colin/mde.git

commit : bd85e2055639643a08e48b6d1768121ec724e821

prompt :

应该要支持有条件的情况下的 直接编辑操作

1 是打开没有staged的diff,就是git的changes

2 需要切换到 左右两栏对比时

结果:

1. **Opus 4.8-xHigh(1M):** 尝试了很多轮对话，就是不能达到比较理想的状态，越改越差
   1. 第一次轮，能在分栏里面进行编辑，但是打开新的 tab+没有脏标记
   2. 第二轮，没啥变化
   3. 随后就越改越差，还是会打开新的 tab
2. **doubao-seed-2-0-code-preview-260215:**
   第一次改完后，连文件都不能打开了，后续再修也完全不沾边
3. **GLM-5.2(1M):**
   1. 第一次轮，额外询问了一些问题后，在分栏里面进行编辑，但是打开新的 tab+没有脏标记
   2. 第二轮，叫我提供日志，还有一些奇怪的文件状态，拒绝后， 196K上下文，非常慢，卡死，和 opus4.8一样，尝试用个守卫逻辑， tab打开后就自动关闭。
   3. 第二轮，竟然成功了，和 Fable采用了同样的方法，用了缓存方案可编辑的 diff 不再使用 applyEdit，虽然细节还有瑕疵，很慢， 但是比 **Opus 4.8**要好，比 **Fable5**要差。
4. **Fable 5-xHigh:** 经过 **3**轮的对话，就完成了所有功能，没有进行提供额外的信息

应该要支持有条件的情况下的 直接编辑操作

1 是打开没有staged的diff,就是git的changes

2 需要切换到 左右两栏对比时

可以编辑，但是 编辑了vscode 会自动打开一个 编辑tab,

并且diff 那里也不显示 文件脏标记

现在比较正常了，唯一不正常的是，在diff 里面编辑，会直接自动保存，不会标记修改，然后等我保存

# Test9

git: `http://deve.work:33333/colin/alldata.git`

commit: `c7255d9e33e416862fed2dd36604da0dcfcd09b3`

prompt:

请使用这个 Auto/FactGraph/认知图: 2.9.fact函数式设计稿.md 生成的方法评估一下 deepseek-v4-pro 和 glm-5.2 两个模型的能力

不要修改Auto/Resource/Fact这个目录的文件，可以生成去临时文件夹再评估

不要关注这些机械性的 问题，要 关注 fact最根本的目标的达成， 语义的抽取。不要用机械校验

结果:

GLM-5.2 更准确的地方

1. 归类关系判断（700.txt @b3）：原文"极少量的参数...仍可能包含数十万权重"是一个归类判断（超级权重是大模型参数的一种）。GLM 写是("超级权重", "大模型参数")——对; deepseek 写 属于("超级权重", "大模型参数")——错，这恰好是设计稿明确举例警示的那个反例（"属于"只表隶属，归类关系该用"是"）。
2. 避免语义拧巴（700.txt @b22,b24）：原文说"避免破坏 LLM 生成连贯文本的能力"。GLM 写 破坏(..., "LLM生成连贯文本的能力")——准确; deepseek 写成 破坏(..., 生成("LLM","连贯文本").极性("否"))，相当于说"破坏了『LLM不生成连贯文本』"，逻辑主宾关系拧了。
3. 保留限定语气（1000.txt @b15）：原文"现有模型多只能生成静态图片"。GLM 保留了"只能"的限定; deepseek 写成.模态("可以")，把"只能做到X"简化成"可以做到 X"，丢了"仅此而已"这层限制意味。
4. 量化词不夸大（1000.txt @b17）：原文是"绝大多数系统"（不是全部）。deepseek 写所有("绝大多数系统")——字面上自相矛盾（用全称量化词"所有"去包裹一个"大多数"的表述）; GLM 没有陷入这个矛盾，保留了"大多数≠全部"的原意。
5. 没有漏掉关键限定词"首次"（1000.txt @b58）：原文"AI 系统首次在金融投研场景中展现出接近人类分析师的能力"。GLM 保留了"首次"; deepseek 这条事实完全没有提到"首次"，是一处实打实的漏译。

deepseek-v4-pro 更准确的地方

1. 不制造额外的必然性判断（1000.txt @b15）：原文只是说现有模型"缺乏严格的数据一致性约束"（陈述现状）。deepseek 老实写 缺乏(...); GLM 却写成具有(...).模态("不可能")（"不可能具备"），把一个描述性的"目前缺乏"拔高成了"根本不可能具备"的必然性判断，属于无中生有的夸大。
2. 独立成行更彻底（700.txt @b6）：原文"大模型变得胡言乱语...只会瞎猜"是两个独立效果。deepseek 拆成两条独立语句; GLM 挤在同一事件下用两个.效果(...) 续接表达——内容没丢，但没有真正做到"独立成行即断言"。
3. 列举内容的结构化程度：这是两个文件里最一致的差异——原文顿号列举的内容（如"股价、财报、新闻""结构化金融数据与非结构化信息"），deepseek 普遍拆成 并列(...) 构造; GLM 经常整句塞成一个自由文本字面量。设计稿明确要求"顿号列出的几个成员...该用并列构造，不能因为省事拍扁成一句自由文本"——这一点上 deepseek 做得更彻底。如果目标是建成可查询的知识图谱而不只是"语法合法的复述"，这个差距是实质性的。

两个模型共同的问题

- 700.txt @b13/b14: 两者都把"原始 Llama-7B"错误地安到了剪枝之后的效果上（原文剪枝前才叫"原始"），这是同一个误读，不分高下。
- "难以"被两个模型都系统性地夸大成了.模态("不可能")（1000.txt @b12↗@b15 两处，两边一致）——"难以"≠"不可能"，这个夸大两个模型都在犯。
- 1000.txt @b13 整句（通用搜索系统难以形成系统性洞察的原因）两个模型都漏译了。

# Test10

git: `http://deve.work:33333/colin/logix`

commit: `5ad99ded353b1a022c6041182c51322564e80551`

prompt:

> 在大的色块显示高亮的时候，还是会有 浮点型 取整的问题，导致右边框偏左一点点，
>
> 请全局检查下，所有的坐标 浮点问题

不能进行任何有用的信息提示，如果 Agent主动问，就说不知道。

看看结果能不能直接解决问题，注意最右边需要特殊处理才行

**答案: **最右边的色块特殊，是因为间隙缝（ gap）的判定不对称。

drawBarLane 只在「后面还有一个够宽的相邻块」时才画那条 2px 白缝分隔（ if (!nx) continue）：

- 中间的块：画缝 → 可见颜色只到 xr - 2
- 最右 /最后一个块：不画缝 → 颜色一直填到 xr

结果:

**Opus 4.8-High(1M):** 没有一次性解决，在提示了 `为什么最右边的色块的现象和其他的不一样`

之后就解决了

**GLM-5.2**:

**Fable 5-High:**

# Test11

git: `http://deve.work:33333/colin/alldata.git`

commit:`52379d92275bcbbfb1a633515edde7a74977ba17`

prompt:

> `/home/colin/develop/AllData/Agent/cogmap/docs/fact改进.md`
>
> 把这里面的30个示例，写成一个测试文件，然后文档里面删除示例，然后 引用这个测试文件

结果:

**GPT-6-Astra high:** 写了一个 markdown文件，我要的是一个 python测试文件

**GLM-5.2:**

**Fable 5-High:**

# Test12

git: `http://deve.work:33333/colin/alldata.git`

commit: `52379d92275bcbbfb1a633515edde7a74977ba17`

prompt:

> 现在在 Agent里面，只有一个web的前端，现在需要增加一个 CLI的前端，
>
> 两个前端是同步的，web可以操作，cli也可以，
>
> cli可以参考 /home/colin/develop/Wiv 的实现，命名也是Wiv，
>
> 放在Agent里面单独一个文件夹 叫cli

结果:

1. wiv都不能直接启动　　cli后台
2. cli后台都不是原来的　　Wiv的界面风格

**GPT-6-Astra medium:** 基本不能直接用，各种错误

**GLM-5.2:**

**Fable 5-High:**

---

Revision #32
Created 2026-03-16 07:45:44 UTC by Colin
Updated 2026-10-09 13:52:53 UTC by Colin