为数据库与预测结果设计存储和GPU服务器方案

完整操作教程 · 参考固定 · 往返恢复验收

用分挂载点、分阶段的共存表计算空间，再把数据库读取、CPU搜索和GPU推理放入同一可验证方案。

运行边界：示例代码需按安装版本和真实项目输入复核；未提供真实生物数据，语法或小样本检查不代表科学结果和性能验收。

数据库、工作盘和GPU应一起安排，但并不是同一个资源问题。搜索库能放下，不代表临时索引和结果也有空间；GPU显存足够，也不保证数据库读取顺畅。先把文件按实际阶段和挂载点列清楚，再验证一条完整任务。

建立文件生命周期表

每项记录组件、版本、所在挂载点、GiB占用与存在阶段。下载包、展开库、索引、旧库、模型、输入、编译缓存和结果分别列行。若某项只有估计，标出依据；未确认的容量先调查，不把空白当零。

阶段应对应真实操作，例如download、unpack、predict和update。一个组件在多个阶段都保留，就在各阶段登记；多个挂载点分别计算，不合成一个看似充足的总盘。共享一份文件时也应避免重复计数。

空间部分	为何需要	切换或清理条件
参考数据库	搜索与模板读取	新发布验证后再切换
独立工作区	索引、搜索与模型中间数据	结果验收并满足留存计划
预测结果与缓存	复核、复现和重复运行	按明确用途保留
更新及归档副本	回滚、迁移和备份	复读校验与切换记录完成

按阶段计算各盘共存上界

准备storage-plan.csv，列mount,phase,component,gib，gib统一使用GiB。下面标准库脚本先求每个挂载点各阶段之和，再取最大阶段；它只计算你填的计划，不自动发现文件或保证实际峰值。

```
import csv, math
from collections import defaultdict

totals = defaultdict(float)
seen = set()
with open("storage-plan.csv", encoding="utf-8", newline="") as handle:
    rows = list(csv.DictReader(handle))
assert rows
for row in rows:
    mount, phase, component = (row[k].strip() for k in ("mount", "phase", "component"))
    value = float(row["gib"])
    assert mount and phase and component and math.isfinite(value) and value >= 0
    key = (mount, phase, component)
    assert key not in seen, "duplicate component within one phase"
    seen.add(key)
    totals[(mount, phase)] += value
for mount in sorted({m for m, _ in totals}):
    stages = {phase: amount for (m, phase), amount in totals.items() if m == mount}
    print(mount, "phase_totals_gib", stages, "planned_peak_gib", max(stages.values()))
```

例如一阶段有下载包与旧库，另一阶段有展开的新库与旧库，两阶段分别相加后再比较。还应另加有依据的剩余空间与文件系统开销，配额和inode分别核对；不能把脚本算出的数值直接当供应方必须承诺的容量。

用代表任务验证读写与计算

在确定的软件和数据库版本下，顺序执行一次搜索、推理、保存与归档，记录各段耗时和工作盘变化。数据库放在远程共享存储时，先测真实读取；不能只依据磁盘标称带宽判断。迁移到本地盘还需计算复制时间和双份共存。

CPU搜索、GPU推理和缓存条件分开记录。当前部分工具提供GPU MSA路线，但它们有自己的数据库与索引要求，不把推理环境的可见显卡自动视为搜索路线也已准备好。设备兼容、参数允许使用与实际容量都要满足。

团队使用时固定版本和写入边界

共同参考使用稳定的版本路径，个人任务写入独立目录；不要让更新脚本原地替换正在使用的数据库。模型参数的访问与共享按对应条件管理，不能因为数据库可共用，就默认参数也能任意复制给所有用户。

正式方案写清数据库发布、最大输入、任务并发、结果留存和迁出窗口。结束前核对结构、置信度和输入对应，归档复读通过后再处理不再需要的工作文件。只有明确缺的是存储或读写能力，再依据这份计划增加相应资源，不把GPU升级当成硬盘问题的处理。

把参考数据库、模型权重和缓存拆开记账

AlphaFold系列任务的存储至少包括遗传/模板数据库、模型参数、输入与MSA、运行中间文件、最终结构和归档备份。每项都记录来源、版本、压缩/解压状态、只读或可写需求、所在挂载点和清理周期。数据库不是“下载后占用大小”这么简单：解压后容量可能显著增加，数据库更新期间新旧版本还会同时占空间。模型权重受使用条款约束，应使用获准位置并限制复制。

AF3的空间和计算阶段有不同峰值

当前AF3官方安装说明提到完整遗传数据库解压后约630 GB，整体磁盘规划可达约1 TB，建议SSD用于数据库访问；数据pipeline还可能需要至少64 GB级别RAM作为规划参考，推理阶段需要兼容GPU。具体需要随数据库、输入规模和运行方式变化，不能只按某个公开基准承诺。拆分pipeline与inference分别测数据库搜索、临时空间、GPU显存和结果写出，可以看出瓶颈发生在哪一阶段。

下载目录不要放在构建上下文里

AF3官方说明建议数据库目录不要放在源码仓库内部，以免构建Docker镜像时将巨大数据复制进镜像。把源码、数据库、权重、输入和输出分开放置，并为容器映射只读数据库及权重、可写输入输出目录。若目录权限不足，外部MSA工具可能给出难以理解的错误；运行前以实际作业用户检查读写权限和可用空间，而不是只用管理员账号验证。

GPU可见和GPU有足够显存是两回事

nvidia-smi能看到设备不代表当前容器、作业或用户有独占权限；还要查看实际作业环境中的CUDA/JAX设备、驱动兼容性、显存余量和其他GPU进程。CPU、RAM、网络和数据库存储同样会限制整个流程。仅增加GPU可能无法改善MSA搜索或模板读取阶段；反之，数据库读取慢也不意味着GPU不够。先用相同输入记录分阶段时长和设备利用率，再决定资源组合。

用分级试跑做存储和并发计划

先单独测下载/解压，记录网络耗时、临时空间峰值与解压后容量；再用官方小样例完成一次完整预测，记录CPU/GPU各阶段资源；最后以代表性真实序列试跑。只有相同数据库版本和相似输入条件下的数据才适合扩展估算。并发前将数据库设为共享只读，输出、缓存和scratch按任务隔离，并保留足够空间给多个失败重跑；监控文件系统配额和inode，而不只看df的全盘容量。

AF3数据库容量规划参考

AlphaFold 3官方安装文档目前给出的完整数据库压缩下载约252 GB、解压后约630 GB，整机磁盘规划可到约1 TB；还需给模型参数、容器镜像、工作缓存、输入输出和备份留空间。实际使用的数据库子集、版本和共享方式会改变占用，以上数值用于初步规划而非对每个环境的保证。建议先核验当前官方安装说明，再从下载清单与实际解压结果计算项目可用容量。

不要把GPU与数据库归为一种瓶颈

数据pipeline需要在本地读取序列库、结构库并执行搜索，通常受CPU、RAM、磁盘延迟及文件权限影响；模型推理则关注GPU型号、显存与运行时兼容。日志中的阶段时间、设备利用率和IO wait能够帮助定位瓶颈。若GPU阶段很快而pipeline等待明显，应先调查SSD与数据库读取；若pipeline结束但推理报显存错，再调整推理任务的输入复杂度或GPU资源。

更新数据库需保留版本切换方案

共享数据库更新期间，先下载到独立版本目录、完成完整性核对和权限检查，再将新版本对用户开放。不要直接在正在运行任务的路径上覆盖数据库文件，避免同一任务前后读到不同版本。迁移或更新后，用代表性样例确认搜索工具能读到目标库并记录实际数据库版本；保留旧版一段时间，以支持项目复跑或出现回归时回退。

备份不是简单复制所有缓存

数据库、模型权重和派生缓存的可重建性及许可不同。根据项目恢复需求决定备份哪些文件，记录缓存生成命令和数据库来源；受许可限制的模型权重按官方条款存储，不擅自打进课题共享包。对可从公开源重建的数据库，可保存版本、下载校验与部署脚本；对昂贵或受限数据则使用机构批准的备份位置并验证恢复权限。

快速读取与长期存储可分层

数据库不一定都要永久放在同一类存储上。结合官方工作流支持的路径选项，可将频繁访问的部分放在本地SSD，把其余数据库放在稳定的大容量存储，并按工具明确检索路径优先级。不要在任务运行时移动或重命名已打开的数据库。若SSD空间有限，先对代表任务测不同路径的阶段耗时；缓存命中、并发用户数和网络文件系统延迟都会影响实际收益。

容器挂载和权限应按作业用户验证

Docker容器内外路径不同，命令中的`--model_dir`和数据库路径必须与实际volume挂载点一致。检查输入可读、输出可写、模型权重可读、数据库只读以及容器用户对目录的访问权限。管理员账号能读而www-data/容器用户不能读是常见迁移差异。启动前用作业身份执行目录读写和空间检查，避免大规模数据库搜索到中途才因权限或配额失败。

计划写入前先测容量和inode

按数据库目录、解压目录、模型目录、容器缓存、临时目录和结果目录分别检查df与配额。下载和解压常会短时同时保留压缩包与展开文件，更新数据库时又可能需要新旧两份。若文件数巨大，还要检查inode额度。先在目标盘做一小段下载/解压或模拟大文件写入，确认路径、权限和余量后再执行全量准备。

更新后的数据库要能回到旧版本

为数据库版本建立只读命名目录，保存官方下载/部署脚本、版本说明和校验清单。用户任务记录所引用的数据库目录版本，更新不改变既有路径内容；需要升级时创建新版本目录并通过小样例验证，再有计划地切换。这样同一项目可继续使用原数据库重跑，团队也能逐步迁移而不是全员被一次更新打断。

适用范围与结果归档

数据库更新需要同时核对版本、文件完整性和运行用户权限。建立只读正式目录后，再由作业用户执行一个小样本搜索，确认数据库可读取；不要只凭文件存在或目录大小判断部署成功。

数据库和模型权重可能包含许可限制，本文给出的容量估算不替代官方条款核验。确定共享与备份位置前，确认项目用途、可访问人员和机构规定，并保存管理员认可的部署说明。

输出目录按任务ID和软件版本隔离，并确认不同GPU作业不会写到同一个结果路径。每批结束后核查结构数量、摘要文件与日志，再按归档策略转移结果；临时缓存清理前确认数据已成功保存。

小结

如果因磁盘不足无法保留全部版本，先确定项目复现所需的最小材料，并保留官方版本信息、下载校验和、部署脚本和最终实际使用的数据库。需要清理旧数据库时，确认没有仍在运行的任务引用该路径。

配套练习包

本PDF对应的ZIP包含本文代码块、主题练习模板或合成数据、README和SHA256校验文件。请从同一教程页面下载练习包；先核对校验值，再在独立目录复现。

参考资料

AlphaFold2官方数据库与预设 · 来源资料

AlphaFold3官方数据库与运行环境 · 来源资料

AlphaFold3输出和可选大型文件 · 来源资料

ColabFold本地MSA与数据库路线 · 来源资料

AlphaFold 3官方安装、数据库和磁盘规划 · 来源资料

操作建议：先在合成小样本上验证流程，再依据项目参考、标签合同、存储策略与获配资源验收真实数据。