常见问题、排障与维护

OpenSAI 常见问题、排障与日常维护

按常见现象快速定位环境、连接、应用依赖与性能调参问题，保障集群长期稳定运行。

产品概览 快速上手 使用手册 智能体接入

排查同一次执行时，建议保留返回的作业号（`job_id`），以便在命令行、Web 控制台与运行日志之间交叉核对。

常见现象速查表

你遇到的现象	常见原因	第一步排查操作
终端提示找不到 `opensai` 命令	当前终端未激活项目虚拟环境 `.venv`	执行 `source .venv/bin/activate`（或 `.venv\Scripts\Activate.ps1`）后重试 `opensai --help`
控制台能打开，但看不到本机设备	当前处于“演示数据”模拟预览，或设备进程未连接中枢	点击顶部横幅 **“切换到实时数据”**（或在 **“设置”** 关闭演示数据），并在终端运行 `opensai devices list`
应用或 DAG 运行返回 `FAILED`	缺少第三方库、本地数据集路径不匹配或步骤代码报错	在控制台 **“任务”（Runs）** 点开失败作业与对应步骤，直接查看详细报错文本
数据文件已拷入机器，但应用报找不到	未放在 `OPENSAI_DATA_DIR` 直接子级，或名称与清单不一致	运行 `opensai data list` 核对设备实际发现的数据集名称
等待结果超时，或 MCP 返回 `done=false`	任务耗时超过客户端单次等待窗口，后台仍在运行	**不要重新提交**；使用原 `job_id` 执行 `opensai result get YOUR_JOB_ID` 继续查询
指定了截止时间，但应用旋钮未发生变化	当前算力足以在默认高档位按时完成，或调用时用 `-k` 锁定了旋钮	运行 `opensai explain YOUR_JOB_ID` 查看候选评估；如需将设备选择纳入联合优化，启用 `adaptive` 模式

1. 安装与首次启动

必须有多台电脑或独立 GPU 才能使用吗?

不需要。单台普通笔记本即可同时运行协调中枢、1 个工作设备进程与 Web 控制台，完成全部基础功能验证。`pi_pipeline`、标准库应用模板、`number_stats`、`cluster_analytics` 与轻量 `coop_perception` 均可纯 CPU 运行；仅在运行 `how2comm_perception` 等深度学习模型步骤时，才需为对应执行节点配置 GPU 与 PyTorch 环境。

需要预先安装 Node.js、Docker 或 Kubernetes 吗?

均不需要。发布包已内置编译完成的 Web 控制台静态资源，开箱即可通过浏览器访问；所有计算步骤均在设备端轻量 Python 进程池中隔离执行，不依赖容器运行时。仅当你修改了 `frontend/` 源码并希望重新构建前端界面时，才需要 Node.js 工具链。

安装时下载 Python 依赖较慢如何加速？

可通过环境变量 `PIP_INDEX_URL` 指定所在网络访问顺畅的 PyPI 镜像源后再运行安装脚本，例如：

```bash
PIP_INDEX_URL=https://pypi.tuna.tsinghua.edu.cn/simple bash scripts/setup.sh
```

基础体验仅需核心环境，无需在首次启动时安装体积较大的机器学习扩展包。

2. Web 控制台与跨机网络连接

控制台页面能打开，但左下角显示“中枢离线”或“已断开”怎么办？

浏览器页面由 `user_client.py`（默认仅本机监听 `127.0.0.1:5000`）提供服务，它再向协调中枢（默认 `TCP 8888`）保持自动重连链路。请先在终端执行 `opensai cluster status`：若终端同样连不上，说明中枢进程（`server.py`）未启动或端口不一致；若终端正常而页面未连接，请检查控制台左下角 **“设置”（Settings）** 页面中的“桥接地址”配置（默认同源留空即可）。

为什么局域网其他电脑打不开 `:5000` 控制台？“设置”里的系统参数为什么不能直接改？

- **默认仅监听本机**：由于 Web 桥接服务不内置访问鉴权，默认仅绑定 `127.0.0.1:5000`。若需在同一可信局域网的其他电脑访问，可在启动前设置 `OPENSAI_BRIDGE_HOST=0.0.0.0`（或使用 SSH 端口转发 `ssh -L 5000:127.0.0.1:5000 <中枢主机>`）。
- **进程级启动参数**：中枢放置策略（`OPENSAI_PLACEMENT`）与中继多链路开关（`OPENSAI_MULTIPATH`）在进程启动时读取，需在环境中设置后重启对应进程生效；**“设置”** 页底部的 **“系统参数”** 列表用于集中查阅当前参数含义与默认值。

跨不同局域网或异地设备如何组网？

建议先通过 Tailscale、WireGuard 或团队受控 VPN 将中枢、客户端与各工作设备接入同一虚拟私网，再将 `OPENSAI_HOST` 设为中枢在该私网中的可达 IP。

> ▲ **安全与边界提醒**
>
> **端口与安全提醒**：除放行中枢控制面端口（默认 `TCP 8888`，无内置公网鉴权，切勿直接裸露在公网）外，跨设备传递中间产物还要求参与工作设备之间的数据面端口（默认 `UDP 9999`，QUIC 协议）彼此互通——仅能单向访问中枢不足以完成跨设备数据拉取。

3. 应用运行、依赖、数据与模型

应用包注册成功了，为什么执行时报 `ModuleNotFoundError`？

`opensai app register` 会自动将 `steps/` 下的步骤源码分发同步至工作设备，但不会替远程机器自动安装第三方 Python 库。请对照 `app.yaml` 中的 `requires` 列表，在所有可能执行该步骤的工作设备 `.venv` 环境中预先安装对应依赖。

模型权重与大型数据集应该放在哪里？

放在实际承担对应读取或推理步骤的工作设备上：

- **数据集**：默认位于 `Offloading-System/data`（或 `OPENSAI_DATA_DIR` 指定的目录）的**直接子文件或子目录**下，设备启动后会自动上报至 `opensai data list`；如果数据存放在大容量外部磁盘，建立软链接指向该目录即可。

- **模型权重**：代码同步不包含大体积权重文件；请按应用自身约定的路径部署在推理节点上（如 `how2comm_perception` 边缘 GPU 节点）。

内置的 `video_counting` 接的是真实摄像头吗？如何换成自己的 YOLO 模型？

内置 `video_counting` 使用 `numpy` 实时生成合成帧并核对已知目标数，方便在无摄像头环境下验证双分支流水线与分辨率/抽帧旋钮。接入真实 RTSP 摄像头、本地视频或自定义 YOLO 检测器时，只需替换采集步骤与检测步骤，在执行设备装好 OpenCV / PyTorch 与权重，将允许平台动态权衡的分辨率或抽帧间隔声明为 `knobs`，将必须固定的业务阈值保留在常规参数中即可。

为什么作业中有节点显示 `SKIPPED`，整体状态依然是 `SUCCEEDED`？

这是 OpenSAI 面向动态边缘环境的容错机制：应用清单可将可能临时离线的协作节点（例如驶离路口的协作车辆）标记为 `optional: true`。当该分支缺数据、设备离线或主动返回 `SKIP:` 时，平台会安全跳过该分支，由下游必需融合步骤利用其余在线源完成计算。

4. 性能指标与自适应行为解读

设置了 `--deadline` 就一定保证不超时吗？

`deadline_s` 是指导调度器筛选设备、分配核数与下调应用旋钮的优化目标，而非打破物理算力极限的硬实时承诺。任务完成后请同时查看 `duration_s` 与 `qos` 达成字段：若在最轻档位下仍持续超时，说明当前硬件或带宽相对输入负载不足，需扩容设备、启用 GPU 或降低输入帧率。

为什么多台设备跑不一定比单机更快？开启多链路后为什么有时仍走直连？

- **多机收益边界**：分布式执行引入了跨机调度与网络传输开销。对于计算量极小、强串行或中间产物巨大的任务，同机本地执行反而更划算；可通过节点级 `compute_s` 与 `fetch_s` 定位瓶颈。
- **按需启用多链路**：开启 `OPENSAI_MULTIPATH=1` 后，`DELTA` 规划器会综合传输体积、建连开销、中继空闲带宽与共享出口瓶颈逐次决策。当数据块较小、无空闲中继或所有路径共用同一饱和物理网卡时，走单路 QUIC 直连本身就是最优解。

为什么 `duration_s` 与各节点 `compute_s`、`fetch_s` 相加不一致？“质量 0.9”代表 90% 准确率吗？

- `duration_s` 是整张任务图从开始到结束的端到端墙钟时间；而 `compute_s`、`fetch_s`、`queue_s` 是所有节点的累计和——当多条并行分支同时运行时，累计时间自然可能大于端到端总耗时。
- 质量得分（`0`～`1`）的物理含义由各应用自行定义（如检测 F1、样本覆盖率或归一化精度），跨配置对比时请在同一应用、同一质量口径下进行。

5. MCP 与智能体接入

已经在客户端导入了 `SKILL.md`，为什么智能体还是没有工具？

`SKILL.md` 是写给智能体的操作方法论，实际可调用的 `opensai_*` 工具由 `opensai_mcp` 服务提供。请先按 智能体接入指南 第 1～2 节安装 `--agent` 依赖并配置 MCP 连接，运行 `python -m opensai_mcp --check` 验证通过后再开始对话。

6. 日常维护、结果归档与版本升级

OpenSAI 的运行态作业记录与设备中间产物设有自动过期清理（`OPENSAI_JOB_TTL_S`、`OPENSAI_ARTIFACT_TTL_S`），不作为永久数据库使用。日常运维建议妥善保留以下资产：

资产类别	建议保管与维护方式
自定义应用包源码	独立纳入 Git 版本控制，保留 `app.yaml`、`steps/` 与版本号记录
数据集与模型权重	存放在代码目录之外的独立大容量磁盘，通过 `OPENSAI_DATA_DIR` 或软链接挂载
启动配置与访问令牌	记录自定义 `OPENSAI_*` 环境变量；将 `OPENSAI_MCP_TOKEN` / `OPENSAI_ADMIN_TOKEN` 保存在受控凭据配置中
关键实验与业务结果	通过 SDK 将 `result.to_dict()` 导出为持久化 JSON，或由步骤直接写入业务归档目录
运行诊断日志	排障时查阅项目根目录 `.run/` 下的 `server`、`device` 与 `user` 日志

升级新版本时，先等待在途作业结束并备份上述资产，将新版代码包解压至独立目录完成环境安装，恢复数据挂载与环境变量后先跑通一次轻量冒烟任务，确认无误再切换生产调用入口。

7. 获取支持与提交问题报告

欢迎通过官方仓库 github.com/INSAI-institute/OpenSAI 的 Issues 提交故障报告，或通过 Discussions 交流部署场景与功能建议。提交问题时请尽量提供：

1. OpenSAI 软件版本与材料版本、操作系统及 Python 版本；
2. 集群拓扑规模（单机 / 多机设备数）与启动方式；
3. 可复现问题的最小应用清单 / DAG 输入、完整报错文本及对应 `job_id`（分享日志前请务必脱敏移除 Bearer 令牌、内网敏感账号与私有业务数据）。

若发现凭据泄露或潜在安全漏洞，请优先通过仓库 Security Advisories 私密报告通道反馈，勿在公开 Issue 中直接披露利用细节（更多协作规范见 贡献与学术引用说明，英文版见 FAQ, Support & Maintenance）。