你的问题涉及两个层面的并行优化策略设计：

1. **针对单个 case（材料系统）的优化过程**，如何并行化 Optuna（贝叶斯搜索）与梯度优化阶段。
2. **多个 case 同时提交时**，如何协调多个优化任务的并行运行，最大化硬件资源利用率。

现在我来基于你澄清的内容，重写并完善这份并行优化方案的技术报告，内容将包括 Optuna 和 Levenberg-Marquardt（LM） 方法的特性、并行方式、调度策略等。

**光谱拟合并行优化方案报告（基于 Optuna + 牛顿法）**

**一、项目背景与技术路线**

我们目前的光谱拟合任务主要通过两阶段优化完成：

**1. 贝叶斯搜索（BSU化）阶段**

* 使用 **Optuna** 进行全局优化，搜索材料模型参数空间。
* 这一阶段每组参数组合都要调用 RCWA 引擎模拟一次光谱结果。

**2. 梯度优化（LMU化）阶段**

* 使用 **L-BFGS** 或 **Levenberg-Marquardt** 等牛顿法变种进行局部优化。
* 以 BSU 的 Top-N 解为起点，进行高精度拟合收敛。
* 梯度计算依赖模型残差，过程难以完全并行。

**二、现有挑战**

| **挑战** | **描述** |
| --- | --- |
| 材料参数不能直接传递，只能通过文件方式写入引擎目录 | 多任务之间不能共用一个引擎实例，存在 I/O 竞态 |
| Optuna 虽支持并行化，但每个 trial 都需调用引擎 | 多 trial 间需要隔离材料文件、独立引擎实例 |
| LM 阶段无法粗粒度并行 | 梯度优化依赖状态收敛，强耦合，需从头到尾连贯运行 |

**三、并行优化架构设计**

针对以上挑战，我们提出两个可组合使用的并行方案。

**方案一：单 Case 内部并行（细粒度）**

目标：提高单个 Case 拟合速度，最大化利用多核资源。

**🧠 Optuna 并行（BSU阶段）：**

* 每个 trial 使用独立的材料参数，在 SQLite/RDB 中记录状态。
* 为每个 trial 分配独立的 **RCWA 引擎实例目录**。
* 使用 Optuna 的多进程或分布式策略（如 optuna.storages.RDBStorage）进行并行搜索。

✅ Optuna 自带并行能力，只需保证每个 trial 的执行环境（材料文件、路径、端口）隔离。

**⚙️ Levenberg-Marquardt 优化（LMU阶段）：**

* 通常不并行；每个初始点串行进行。
* 可微调并行策略，如：
  + 多个起点同时开始一个 LM 优化（例如 3 个 Top 结果）
  + 每个起点对应一个 worker，引擎目录隔离

⚠️ LM优化中，每一步依赖前一步结果，不适合细粒度并行；建议以“多个起点”的方式轻度并行。

**🧱 架构图（单 Case 并行）：**

+-----------------------------+

| Optuna Study |

| (SQLite / RDBStorage) |

+-----------------------------+

| | |

[Trial1][Trial2][Trial3] ...

| | |

[Engine1][Engine2][Engine3] (独立路径)

|

汇总结果 → 选择 Top-N → 逐个 LM 优化

**方案二：多 Case 并行（粗粒度）**

目标：在多个材料系统同时拟合时，提高整体系统吞吐量。

**🚀 并行策略：**

* 每个 Case 绑定一个完整的引擎副本 + 材料路径。
* 任务调度器维护一个 RCWA 引擎池，按需分配 Case。
* 每个 Case 从 Optuna 到 LM 完整运行在一个线程中。

**🎯 优势：**

* 任务之间完全隔离，风险低，调度简单。
* 易于水平扩展，支持容器化部署（如 Docker + K8s）。

**架构图（多 Case 并行）：**

+-------------+ +-------------+ +-------------+

| Case A | | Case B | | Case C |

| [Optuna + LM]| | [Optuna + LM]| | [Optuna + LM]|

+-------------+ +-------------+ +-------------+

| | |

[Engine-A] [Engine-B] [Engine-C] <- 独立实例

**四、调度与部署建议**

**✅ 引擎副本部署策略**

* 每个引擎副本包括：
  + 材料配置模板
  + 光谱计算脚本
  + 执行入口（带独立路径/端口）
* 可使用模板机制快速复制 N 份

**✅ SQLite 任务队列管理（用于 BSU 并行）**

* 使用 SQLite 中的 trial 表状态列（RUNNING, WAITING, COMPLETE）实现轻量并行控制
* 各 worker 查询 trial 状态 → 锁定任务 → 修改状态 → 开始运行

**✅ 错误与恢复机制**

* 每个 trial 执行失败应自动记录状态、log、错误码
* 定时任务重试失败 trial 或人工排查

**五、总结与推荐实践**

| **项目** | **方案一：单 Case 并行** | **方案二：多 Case 并行** |
| --- | --- | --- |
| 并行粒度 | trial 级（Optuna） | case 级 |
| 适用场景 | 单个复杂光谱任务 | 多个简单 case 批处理 |
| 引擎副本数 | N（由 CPU 核心决定） | 每个 case 1 份 |
| 实现复杂度 | 中等偏高 | 中等 |
| 依赖组件 | Optuna RDBStorage、SQLite | Python 多进程、队列 |

**六、下一步建议**

1. ✅ 首先搭建 **Optuna + 多引擎副本** 的试验环境
2. ✅ 验证 SQLite + 引擎路径隔离是否满足并发要求
3. ⏳ 后续添加容器化支持（Docker/K8s），实现多 case 弹性部署
4. 🧪 尝试基于 Dask 或 Ray 实现分布式优化队列

如果你希望我继续帮你输出：

* ✅ 技术方案PPT（用于组会汇报）
* ✅ markdown / Word 技术文档
* ✅ 架构图（流程图、组件图）

都可以告诉我，我可以直接为你生成。需要哪种格式？是否需要图示？