工业软件研发中处理超大模型

原创 邓子平 多物理场仿真技术

在一篇文章入门仿真软件性能优化一文中介绍了提升仿真软件性能的一些原则和方法，本文再探讨一下超大模型前处理性能优化方面的问题。

如何定义超大模型?

超大模型主要从两个方面定义，一个是几何单元的数量，一个是网格的数量，分别对应几何复杂度和仿真复杂度。

继续细分的话，考量几何单元数量的同时还需统计每个单元具体的面，边，点等数据以及曲面，曲线的数量。相同的面，参数平面和曲面的数据量可以相差数量级。

分析单元网格数量也要考虑求解模型类型，比如边界元，矩量法满秩矩阵数量会比稀疏矩阵多出几个数量级。相同网格数量，高阶网格自由度会随着会成倍增加。使用显式方法无需求解方程组，非常适用分布式计算且没有误差。

有些复杂的三维几何在结构计算中通过分析，可能将整体简化成计算的二维或者一维单元，在局部使用三维单元，分步分析，简化模型减少计算量。

假设现有一个几何单元数目在10万，六面体网格自由度在1亿左右的模型。

关于数量级的分析，参见“一亿”是“一千万”的十倍吗?

在操作这种模型的时候，不管是GUI操作，视图显示，模型编辑，都非常容易卡顿，或者程序容易死机。工程师对操作这种规模的模型应该是有概念的。

从以下几个方面考察:

1. UI层面：将10万个单元挂在UI树形节点上，如果只是单纯的添加，性能应该没问题，如果涉及到快速反复插入，删除，更新，遍历等操作时，UI会出现明显卡顿。

2. 渲染层面：10万个几何需要单独处理的话，需要10万个实体几何对象，离散成面片，根据实际情况，面片数普遍在千万级别，如果不使用任何加速方法，显示会非常卡顿。刚推出的UE5使用NANITE已经能支持十亿面片的渲染。

对于工业仿真软件的渲染显示，考虑性价比，其加速在已有框架的基础上，更多的是在偏业务层实现。

3. 模型编辑

模型编辑设计到模型的创建，删除，修改，赋属性等操作。

以最常用的创建模型的捕捉操作为例。捕捉操作是为了在模型创建时保证精准性。

比如为了做接触分析，需要创建两个相互接触的对象，第一个对象创建好后，创建第二个对象时需要捕捉到第一个对象的几何。在捕捉第一个对象时，需要遍历第一个对象的几何数据。如果有10万个对象，那就需要遍历10万个对象，因为捕捉操作是实时计算，所以必需要做过滤优化处理。类似操作还有拾取，框选，高亮等操作。

超大模型的几何编辑一定要通过优化过滤等操作控制在局部范围。

4. 模型检查遍历

公差中常见的几何干涉分析，需要遍历计算整个模型中的几何数据。针对10万个三维对象每两个做干涉计算，程序直接拉胯。同理还有分析整体模型中的自由实体，最短边，最短距离，最短形心距，最短质心距。这些计算操作都涉及到整体模型的遍历检查，有些可能是实时操作。

5. 常见I/O操作

虽然现在硬盘读写速度已经达到500M/秒，但超大模型的读写，传输仍然存在瓶颈，尤其是文本文件的读写，优化的空间还很大。

6. 1亿自由度的仿真计算方法后续介绍

如何处理超大模型

1. 提高模型处理的并发性

目前，分布式，并发等技术和工具已经非常成熟。熟悉相应的工具即可应用在程序中。对于研发来讲，其核心仍然是如何将模型分解分治，方便软硬件的并发处理。

2. 开发调试工具

工欲善其事必先利其器，对于超大模型需要开发出相应的调试工具，不能依赖于原始的简单调试方法。

比如在千万级别的网格中，如何快速定位最小网格的材料信息，简单的单步调试不仅需要修改代码，而且性能也无法接受，可能半天时间只够跑几次。

通常软件产品中的性能问题和业务紧密相关，但这些业务相关的内容很难重现，或者重现成本太高。我们可以在调试工具中对业务进行模拟，降低调试开销。

所以，“调试工具”的开发是一项“磨刀不误砍柴工”的工作，虽然并不能直接体现在软件产品上，但却是提升超大模型开发效率的必备工具。

3. 根据数据特征，选择合理的数据结构和操作算法

这个涉及到软件研发中各种数据模型，一方面要精通软件研发中常用的数据结构和算法，另一方面对业务数据和逻辑也要比较熟悉。是一个最能体现研发人员研发功底的地方。细节问题后续再讨论。

4.轻量化底层组件

在工业仿真软件中，会使用各种各样的第三方库或者组件。这些第三方工具往往通用性较强，但并不适合专业的业务。所以可以考虑开发业务相关的底层模块，这样可以将业务做针对性的优化。

5.解耦性能瓶颈功能

利用各种性能查看工具，找到性能所在点。将各种造成性能瓶颈的问题抽象提炼出来，做成专用模块在外部处理。比如海量数据查询检索等，可以交给专业数据库。

6.模型提取和二次优化

通常超大的原始模型中，可能我们只对部分数据感兴趣。可以根据需求提取模型相关数据。对数据进行再加工，简化和重组，有点类似模型降阶技术，方便在定性分析充分之后再做定量分析。

超大模型处理的性能问题符合典型的“木桶理论”：一只木桶能盛多少水，并不取决于最长的那块木板，而是取决于最短的那块木板，“木桶理论”也可称为“短板效应”。软硬件中任何一个短板或者bug都可能影响整体，针对超大模型，一般模型不起眼的问题也可能成为大的瓶颈。

需要说明的是，超大模型的处理也是考验软件架构设计的一个指标，好的软件架构能帮助定位发现问题，也方便功能扩展改进，在软件设计之初需要考虑到。