Structured ALE 与超大模型

陈皓

Livermore Software Tech Corp, 7374 Las Positas Rd, Livermore, CA 94588, USA

摘要： 本文介绍了 LS-DYNA 的 Structured ALE (S-ALE) 求解器在超大规模模型上的应用。

关键词：S-ALE, 超大模型

1 S-ALE 求解器

过去二十年来，ALE 模拟的工程问题类型发生了很大的改变。首先，ALE 模型的规模急剧增长。十六年前，当作者刚刚开始从事 ALE 开发的时候，典型的 ALE 题目包含几十万单元，用 8 个或十六个 CPU 运行。现在，1 千万的单元数量已经非常普遍；CPU 数目也至少三四十个起跳。第二，模拟问题的类型也彻底变了。20 年前，大家一般使用 ALE ELFORM=5 计算畸变很大的拉格朗日固体。而现在，大多数，如果说不是所有的，ALE 用户都使用 ALE 多材料单元（ELEFORM1），来模拟多种流体在 ALE 网格中的流动。最后，ALE 网格也变化了。过去，很多网格创建时，都让网格的边界和物质的表面相吻合；而现在超过 90% 的模型使用方方正正的，规则的 ALE 网格。

这些变化促使 LSTC 开发一个新的，单独的求解器来解决使用规则网格的 ALE 多流体问题。我们的想法是，结构性网格在逻辑上的规律性，可以使得算法的简化，内存使用的减少和效率的提高成为可能；而这些在处理不规则网格时是不可能做到的。2015 年，LSTC 推出了新的结构（规则）网格求解器来针对性地求解使用规则网格的工程问题。我们很高兴得看到，针对规则网格的简化和提高确实符合我们的预期。对同样的问题，S-ALE 求解器效率有很大的提高。

2 S-ALE 处理超大模型的优点

S-ALE 求解器可以更好地处理大规模问题。首先，网格是在求解器中生成的，而不是由输入文件读入的。这样，输入文件中，几何信息和网格拓扑（单元所含节点编号）从可能占据 Giga 字节变成了几乎不占任何储存空间。之前，我们先要 1）在前处理器中生成网格；2）储存网格到输入文件；3）LS-DYNA 读入网格，开内存并将网格数据存入内存。现在，S-ALE 仅仅读入很少的信息，包含：三个方向的网格间隔，起始点坐标，以及局部坐标系。接下来，它会自动生成网格并储存。这一改变带来了极大的时间和储存上的减少，使得处理大问题更为简便。

其次，网格的规律性还可以极大减少内存的使用。在求解器中，几何和拓扑信息以及记录数组占用很多内存。S-ALE 开发者在减少内存使用上下了很大功夫；同时也尽量对 LS-DYNA 总体数据结构不做大的改动。这一减少内存的努力，仍然是一个尚未完全结束的任务。

最后，S-ALE 和 LS-PREPOST 的开发者们一起，也在更好地进行后处理上做了很多努力。我们的想法是，利用网格的规律性，在求解器方面，开发一个新的，更为简洁的后处理数据库；在 LS-PREPOST 方面，优化算法，减少内存使用。

3 内存是瓶颈

运行大模型的主要困难在于内存的限制。在这里请注意，因为大模型必须在 MPP 上运行，所以本文的讨论中只讨论 MPP 的情形。运行 LS-DYNA MPP 时，最大的内存需求一般发生在 INITIALIZATION 过程中的 PHASE 1。PHASE 1 的主要工作是：机器的 head node（0 号 processor) 读入模型并创建数据库储存。在 PHASE 1 之后，大的完整的模型被其拆分（PHASE 2），变成很多个单独的小模型并写成独立的多个输入文件（PHASE 3），然后各个 processor 再分别读入各自的输入文件（PHASE 4）。我们可以看到，在 PHASE 1 中我们需要储存整个模型信息，所以内存需求很大。当然，最简单的解决方法是在 head node 上装巨大的内存，其它 node 上配正常内存。但是这种方式既不现实也不经济。另一种方式是找一个配有巨大内存的单机，在这个单机上做读入模型以及拆分并储存拆分文件，然后将模型输入文件以及拆分文件移到并行机上。在并行机上运行时，程序会跳过 PHASE 1 和 PHASE 2，直接由各个 processor 读入并建立各自的小模型。这一过程比较繁琐而且复杂，即使很有经验的老用户也要试验几次才能成功。

4 减少内存

S-ALE 可以顺利处理现有模型。现阶段，典型模型单元数量从几百万到一两千万不等。但是，S-ALE 的开发者意识到，随着计算能力的快速提高，在可预期的将来，模型单元数量会快速增长。为了让 S-ALE 更好的迎接将有的挑战，作者决定重新构造 S-ALE 的 INITIALIZATION 过程以节省内存的使用。

想法很简单，既然最大内存需求发生在 PHASE 1，那么我们必须试图在 PHASE 1 中减少内存。对于不规则网格来说，这可能相当困难。可是对于规则网格来说，就相对容易地多了。既然网格是由 S-ALE 求解器自己构建的，那么在哪个阶段构建完全可以我们自由决定。那么为什么我们还需要 PHASE 1 呢？我们在 PHASE 4 再构造 S-ALE 网格，岂不是所有问题迎刃而解？每个 processor 只拥有，例如百分之一的模型；顺理成章的，内存需求也就降到原来的百分之一了。

今年年初，作者将这一想法付诸实施。将整个 S-ALE 的 INITIALIZATION 过程重组之后，S-ALE 网格只在 INITIALIZATION 的最后才产生。在那时，每个 processor 只拥有各自的小模型，从而内存需求变得非常少。

5 一个两亿模型的例子

作者选了一个很小的含有三十八万七千个单元的穿甲模型并将其在 3 个方向加密 8 倍，得到了一个 1 亿 981 万 4 千 4 百个单元的模型。输入文件和问题描述可以在如下网址下载：
http://ftp.lstc.com/anonymous/outgoing/hao/sale/models/cthrod/ 执行文件使用 mppdyna dev 134214 版本。运行在一个含有 12 个 node，每个 node 含 12 个 core，共 768 个 core 的并行机上。Processor 是 Xeon(R) CPU E5620，每个 node 配有 100GB 内存。

一共运行了五种情况，分别使用 36，72，144，288，576 个 core。运行时间和 MPP convergence 的结果如下:

# of cores	36	72	144	288	576
Total time(s)	84473	45711	32113	17738	9738
Time per cycle (s)	10.056	5.4405	3.7014	2.0785	1.1761

Speedup	1.0	1.8484	2.7168	4.8381	8.5503