0.1 Методика

Первым этапом является изучение пакета и бенчмарка, их особенностей и возможностей, а так же прикладной части моделирования, так как понимание базовых вещей, на которых строится выполняемая работа может ее значительно облегчить в дальнейшем. Например, в моем случае, в том числе важным фактом для возможности оптимизации был год создания бенчмарка - 2013; Логично предположить, что с того времени было произведено большое количество программно-аппаратных обновлений и улучшений, и нужно знать, каким образом их можно интегрировать в бенчмарк, а так же сборка пакета. По поводу сборки я могу предложить следующие рекомендации, помимо базовых, указанных в документации GMXa:

- 1. Ознакомьтесь с аппаратной составляющей вашей ЭВМ - от этого зависит многое;
- 2. Используйте ccmake при сборке - намного удобнее выставлять ключи;
- 3. Советую начать с тестирования на видеокартах nVidia - они наиболее приспособлены к работе с GMX, и имеют свою проверенную реализацию параллелизма - драйвера CUDA;
- 4. Обратите внимание на архитектуру процессора и количество его физических ядер, поддерживающих набор команд AVX-512 - если их больше единицы, тогда имеет смысл провести сравнение сначала двух-четырех потоков с AVX2-256, а затем, если AVX-512 покажет себя лучше - попробовать на большем количестве потоков. Зачастую, AVX-512 добавляет больше вреда, чем пользы;
- 5. Интересным флагом является `GMX_NO_NONBONDED` - он позволяет получить некоторое приближение расчета симуляции с GPU, не считая при этом ее на GPU - просто отключая несвязанные взаимодействия, и предполагая, что пока CPU считает то, что GPU не в состоянии посчитать - GPU успеет посчитать несвязанные взаимодействия. Это может быть полезно при загруженности кластера, а так же при сравнении с результатами при рельном подсчете на GPU;
- 6. `GMX_NB_MIN_CI` - может быть полезен при работе с небольшими системами, ускоряет вычисления, но только при работе с GPU.
- 7. Почти никогда вам не понадобятся числа двойной точности(double precision), а их использование приводит к деградации производительности, так что следите за этим.

- 8. Существует так же некоторое количество флагов, которые могут отключить внутренний учет и профилировку GMX, их я перечислять здесь не буду - они легко ищутся вот тут -

Второй этап - подготовка предварительного набора файлов для тестировки, их сборка в исполняемые файлы, проверка на корректность сборки в виде пробного запуска на одном MPI потоке. Под предварительным набором файлов я подразумеваю различные настройки силовых полей и версии силовых полей, которые можно создать исходя из здравого смысла и опыта.

Третий этап - это объемное тестирование. Советую проводить на MPI потоках, и следить за тем, что бы GMX самолично не изменял количество OpenMP потоков, например с помощью ключа -ntomp. Количество MPI потоков должно изменяться по степеням двойки, для сохранения баланса между временем и достоверностью. При добавлении еще одного кластера в тестирование, можно добавить и OpenMP. Даже если не использовать все ядра процессора, желательно что бы они все были свободными, как и GPU, если на них считают тоже, иначе это может привести к искажению результата. Стоит использовать равное количество GPU и ядер CPU, иначе это приведет к деградации производительности. Использование метода поиска соседей Верле при использовании только CPU снижает эффективность работы почти в 2 раза по сравнению с методом групп, но для использования метода групп надо совершать откат до 5 версии GMX

Четвертый этап - анализ. Построение графиков, которые помогут сравнить данные друг с другом, и, возможно, увидеть какие-нибудь странности - например, что на более мощном процессоре эффективность вычислений меньше, чем на процессоре послабее. Это уже станет поводом к применению профилировщика и более тщательному расследованию того, что вызвало несовпадения с ожиданиями.

Затем, при выявленной необходимости стоит повторить этапы 3-4, добавив к ним профилировщик, и, наконец, сделать выводы.