Увеличение масштаба неявного анализа LS-DYNA®

Клив Эшкрафт[2], Джеф Доусон[1], Роджер Граймс[2], Эрман Гюлерюз[3], Сеид Корик[3], Роберт Лукас[2], Джеймс Онг[4], Франсуа-Анри Руэ[2], Тодд Саймонс[4] и Тин-Тин Чжу[1]

Крей Инк.[1],
Ливерморская корпорация программных технологий (LSTC)[2],
Национальный центр суперкомпьютерных приложений (NCSA)[3],
Роллс-Ройс[4]

Перевод В. Б. Литвинова под редакцией Б. Г. Рубцова

Резюме

Крей, LSTC, NCSA, и Роллс-Ройс создали партнерство для исследования будущего неявных вычислений, поскольку происходит увеличение и конечно-элементных моделей, и систем, на которых они рассчитываются. Роллс-Ройс создал ряд моделей макета двигателя при помощи конечных элементов, и общее число степеней свободы в этих моделях доходило до 200 миллионов. NCSA запускал расчет этих моделей при помощи специальных вариантов программы LS-DYNA, созданных в компании Крей, на своей системе Блю Уотерс, которая является гибридной системой компании Крей XE/XK с 360 тысячами ядер AMD. Узкие мест обработки и памяти стали видны, когда число процессоров выросло на порядок величины по сравнению с привычным для сегодняшних разработчиков и пользователей, а LSTC вводила улучшения в программу LS-DYNA. В докладе обсуждаются встретившиеся проблемы, улучшения, сделанные вп рограмме LS-DYNA, и результаты, полученные, когда ограничения были раздвинуты, как по масштабу модели, так и по числу процессоров. Это продолжающаяся работа, и мы закончим доклад обсуждением дальнейшей дороги, осветившейся проделанной работой.

Введение

Лишь ограничения по времени и ресурсам могут ограничить размеры и сложность задач на неявный анализ, которые бы хотели решить пользователи LS-DYNA. В силу этого Крей, LSTC, NCSA и Роллс-Ройс образовали партнерство для исследования будущего неявного анализа на предмет увеличения масштабов как конечно-элементных моделей, так и систем, на которых ведутся расчеты. В этой статье мы рассказываем о встреченных трудностях, о том, как некоторые из них были преодолены, и о полученных уроках. Этот продолжающийся проект не мог бы быть выполнен ни одной из участвующих организаций, поскольку каждый партнер привносит в проект уникальные навыки и активы. Хотя многие проблемы с масштабированием были известны разработчикам LS-DYNA, и ими просто не занимались в силу более высокого приоритета других вопросов, но были обнаружены и неизвестные проблемы, которые проявились только при обработке сотен миллионов уравнений на тысячах процессорных ядер.

Статья начинается с обсуждения наиболее актуальной проблемы, с которой мы встретились, а именно с необходимостью переупорядочивать исключительно большие разреженные матрицы с целью сокращения объема хранения разбиения на простые операции и самих этих операций. Это привело к многим другим изменениям в LS-DYNA, а также к открытию удивительных дефектов в хорошо известном коде. Далее мы представляем первоначальные результаты, полученные при прогоне меньшей версии модели макета двигателя в 105 млн степеней свободы смасштабированной на 16 384 ядер на Блю Уотерс. Эта работа показала нам три узких места масштабирования, ранее неизвестных, с которыми мы сейчас работаем. Наконец, мы обсуждаем сегодняшнюю работу, планы на будущее и подводим итоги.

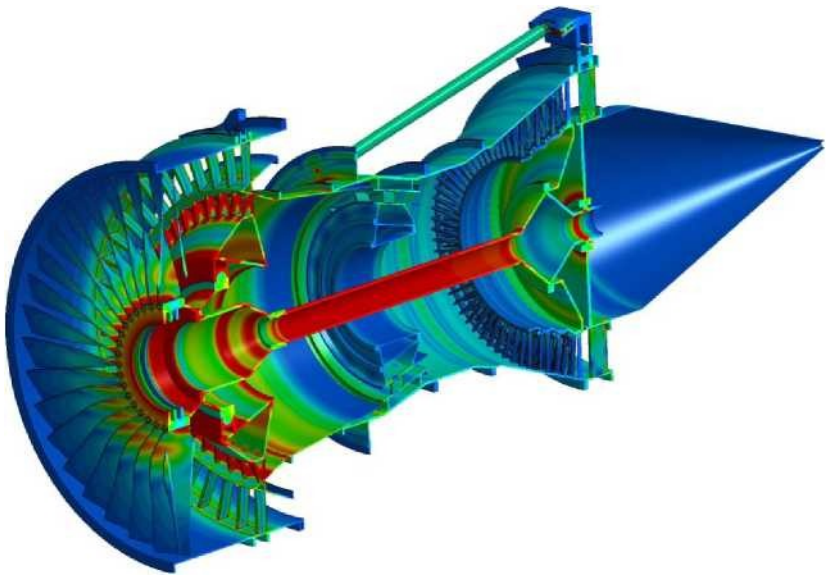

Рисунок 1
Поперечное сечение модели макета двигателя Роллс-Ройс

Проблема с переупорядочиванием

Роллс-Ройс создал набор моделей макетов двигателя, которым может делиться с коллегами по совместной работе в целях расширения границ неявного анализа с точки зрения как масштаба модели, так и масштаба вычислительной системы. Начальный неявный расчет нагрузки с большой моделью (200 млн степеней свободы) был выполнен на внутреннем сервере компании Роллс-Ройс. Метод переупорядочивания по умолчанию, алгоритм вложенного разбиения [George 1973] «Метис» (Metis) [Karypis 1995], не сработал, поэтому вместо него использовали алгоритм «множественная минимальная степень (multiple minimum degree, MMD) [Liu 1985]. Получившееся задание выполнялось вне оперативной памяти системы, для выполнения потребовалось 160 часов на 16-узловом, 224-ядерном кластере с ОС Linux. Хотя это было этапным достижением с точки зрения масштаба модели, время обработки модели было слишком большим, и по этому Роллс-Ройс обратился к коллегам (Cray, NCSA и LSTC), чтобы продемонстрировать использование системы Блю Уотерс с 360 тыс. ядер таким образом, чтобы такого рода расчеты выполнялись за более короткое время, подходящее для инженерных расчетов.

Вложенное разбиение обычно является более эффективным методом разбиения больших линейных систем, используемых в неявном конечно-элементном анализе и LS-DYNA использовала алгоритм «Метис» в течение многих лет. Поэтому сначала мы попытались при помощи Метиса переупорядочить меньшую модель макета двигателя (105 млн степеней свободы) на системе Блю Уотерс. Задание аварийно закончилось, и то, как оно аварийно закончилось, показало, что было переполнение памяти на узлах в 64 Гбайта системы Блю Уотерс.

Метис динамически распределяет доступную память, а LS-DYNA исторически выделяла память, используемую для линейного решателя, статически, и большая часть этой памяти не использовалась в то время, как работал алгоритм «Метис». Для повышения эффективности путем разрешения использования одной и той же памяти и для переупорядочивания, и для линейного решателя, компания LSTC модифицировала LS-DYNA так, чтобы память динамически выделялась и имелось свободная память при неявном анализе. Чтобы можно было осуществлять освобождение и новое выделение больших объемов памяти, бОльших, чем могут существовать в одном процессорном ядре, любое существующее состояние должно быть временно сохранено в специальном файле (scratch file – буквально файл для заметок). К своему большому удивлению мы обнаружили, что такие файлы становятся слишком большими и система не всегда может прочитать то, что сама записала. Чтобы справиться с этим, мы должны были

принять стратегию записи и чтения из блоков небольших размеров. С целью более эффективного использования памяти, впоследствии динамическое выделение памяти было выпущено для общего использования в LS-DYNA версии R11.

Полагая, что ограничение состоит в размере доступной памяти, мы модифицировали LS-DYNA так, чтобы можно было устанавливать контрольные точки упорядочения на одной машине, а потом их считывать на другой. Исследователи NCSA затем попробовал запустить LS-DYNA, с использованием алгоритма «Метис» для переупорядочивания, на системе Бриджес Питтсбургского суперкомпьютерного центра для использования узлов памяти по 3 терабайта. К сожалению, здесь тоже ничего не вышло.

Исследователи в корпорации LSTC в течение долгого времени исследовали LS-GPart, альтернтивную стратегию вложенного разбиения, основанную на множествах (наборах) половины уровня (halflevel sets) [Ashcraft 2016]. Далее мы обратились к этой стратегии, интегрировав экспериментальный код в LS-DYNA. После исследования большого пространства параметров, мы, в конце концов, остановились на эффективной эвристике, результаты которой хорошо конкурируют с результатами алгоритма «Метис». Было очень обидно, но программа работала со всеми тестовыми моделями, с которыми она тестировалась, за исключением большой модели макета двигателя. С большим трудом мы определили, что генератор случайных чисел, которым пользовался наш код ORDRPACK использовал для сжатия неразличимых вершин на графике матрицы, имел период менее 200 млн. Когда мы написали новый генератор случайных чисел, мы наконец смогли использовать LS-GPart для переупорядочения даже самой большой из моделей макета двигателя, как показано на Рисунке 2, на котором изображена модель макета двигателя, разделенная на 8 областей. .

И проблема с генератором случайных чисел, и несогласованность записи/чтения файлов — это примеры того, как хорошо известное программное обеспечение отказывает при работе на масштабах, непредставимых для людей, писавших эти программы, что зачастую происходило десятилетия назад. Есть опасение, что по мере того, как в будущем модели будут расти, проявится намного больше таких проблем.

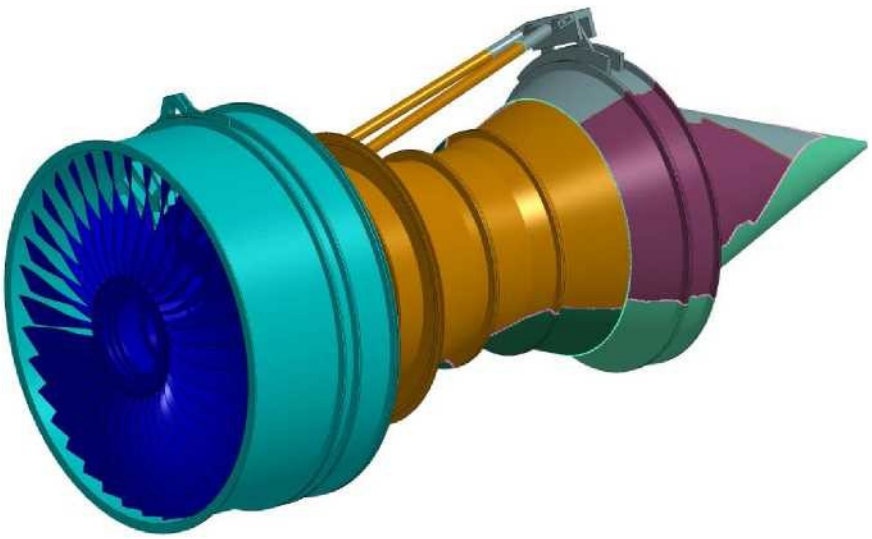

Рисунок 2
Разбиение модели макета двигателя Роллс-Ройс на восемь областей

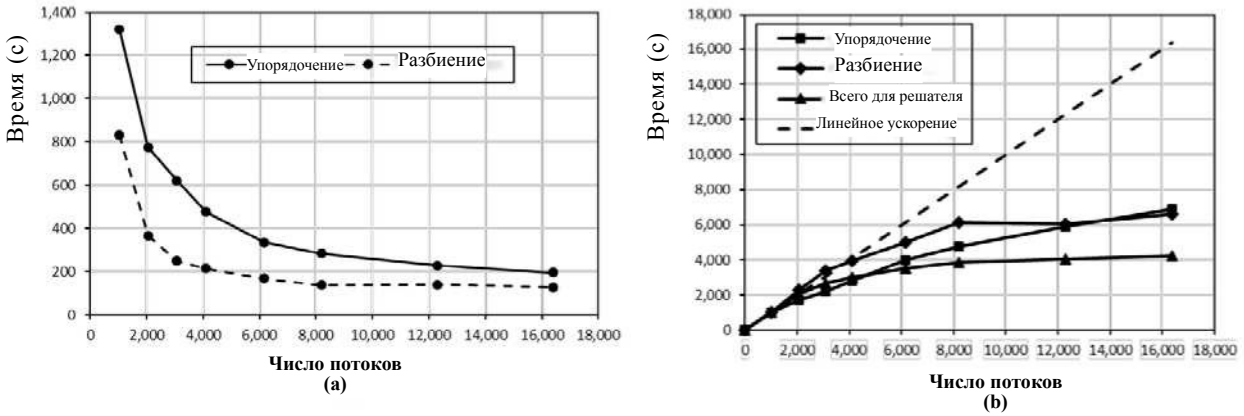

Рисунок 3
Масштабирование переупорядочивания и разбиение разреженной матрицы в зависимости от числа потоков на Блю Уотерз

Начальные результаты, полученные на малой модели двигателя

Когда LS-GPart был эффективно интегрирован в LS-DYNA, мы смогли выполнить расчет малой модели макета двигателя на Блю Уотерз и начать анализ масштабирования LS-DYNA. На Рисунке 3a показано время, расходуемое двумя наиболее затратными по времени аспектами расчета, а именно переупорядочивание модели двигателя с 105 млн степеней свободы при помощи LS-GPart и затем разложение разреженной матрицы при помощи мультифронтального линейного решателя компании LSTC. В расчете было 8 потоков на позицию (rank) MPI, и на графике по оси x показано общее число использованных потоков, которое равняется числу ядер. При начале работы LS-GPart расходуется довольно много времени для переупорядочивания кода, но LS-GPart хорошо масштабируется, и будет масштабироваться еще лучше, когда будут добавлены директивы OpenMP. График производительности мультифронтального кода становится плоским, когда число потоков превышает значение в 8000. На Рисунке 3b показано их относительное масштабирование, нормализованное на 1024 потока. Число в 16 396 ядер — это тот масштаб, которые намного превышает число, доступное для разработчиков в LSTC, или аналогичных небольшоих разработчиков программного обеспечения, так что разработчики таких кодов никогда раньше не имели возможность пронаблюдать работу кодов на этом масштабе, не говоря уже об оптимизации под этот масштаб.

Учитывая наш предыдущий опыт в масштабировании аналогичного мультифронтального кода [Koric 2016], мы настроены оптимистически, и считаем, что наш код может быть существенно улучшен.

Как мы говорили выше, потребление памяти является самой главной заботой. Центр NCSA разработал интрсумент, работающий в режиме реального времени, который работает как фоновый процесс и опрашивает все узлы Блю Уотерз, чтобы определить, сколько памяти доступно. Это позволило LSTC идентифицировать области максимального использования памяти в LS-DYNA. Примеры приводятся ниже, на Рисунке 4, на котором показано, как память выделяется или освобождается по мере того, как LS-DYNA переходит от одной стадии расчета к другой. Представленные результаты относятся к 2048, 4096, 8192 и 16384 потокам, по 8 потоков на позицию MPI.

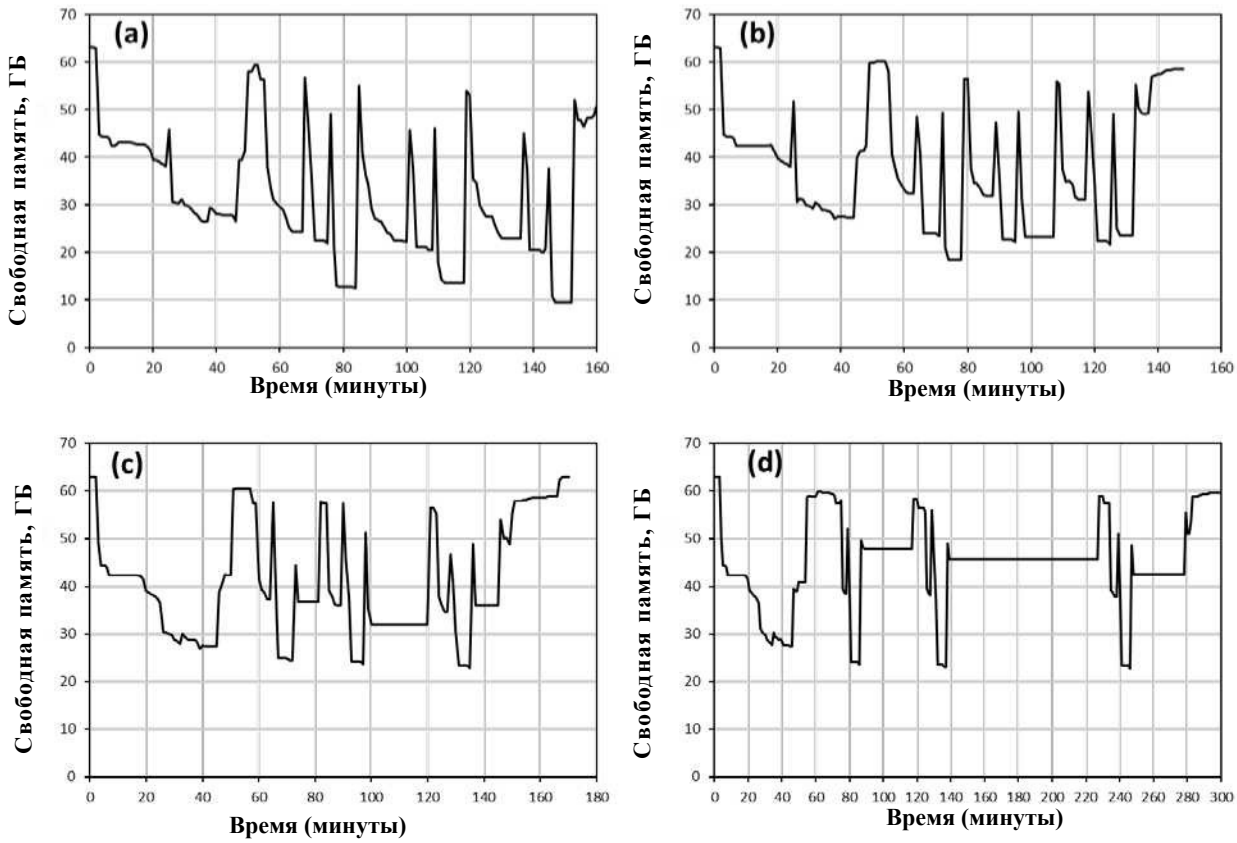

Рисунок 4
Доступная память в зависимости от памяти при расчете малой модели макета двигателя на Блю Уотерз. Общее число использованных потоков: 2048 (a), 4096 (b), 8192 (c) и 6384 (d). На узлах Блю Уотерз по две позиции MPI, у каждой по 8 потоков.

Прогон на 16 384 потоков на Рисунке 4d показывает влияние трех последовательных узких мест: обработку входных данных в позиции MPI 0 (первые 60 минут), символьное разбиение (минимумы доступной памяти) и обработку ограничений (три плоских плато). Желательно проводить обработку входных данных офф-лайн, однако эта программа не соответствовала (не успевала за) изменениям в коде неявного счета в LS-DYNA, и ее нужно было обновлять. В будущем это не будет серьезным узким местом при крупномасштабных параллельных расчетах, когда в течение часа тысячи ядер будут в простое. Символьное разбиение в течение долгого времени выполнялось в LS-DYNA последовательно, обоснованием чему служил тот факт, переупорядочение в «Метисе» тоже выполнялось последовательно и почти всегда занимало заметно больше времени. Масштабируемость LS-GPart привела к тому, что время символьного разбиения стало превышать время переупорядочивания, и в настоящее время активно ведется разработка масштабируемого символьного разбиения. Наконец, «плато» на Рисунке Figure 4d относятся и к обработке ограничений. В настоящее время это проводится последовательно, и процессорное время на обработку ограничений (и анализ, и вычисления) ранее были шумом (пренебрежимы) в общем времени работы LS-DYNA. Однако, по мере того, как мы увеличиваем число позиций MPI до 1024 или более на системе Блю Уотерз, этот шум становится оглушающим (очень значимым).

Текущая работа с большой моделью двигателя

Когда LS-GPart был интегрирован в LS-DYNA, и мы смогли избавиться от проблем, которые мы

имели с алгоритмом «Метис», мы вернулись к большой модели макета двигателя. Попытка запустить эту модель на Блю Уотерз провалилась из-за проблемы с выделением памяти. Чтобы понять, сколько необходимо памяти, компания Крей предоставила для тестирования систему с новыми узлами, содержащими по 192 ГБ памяти на узел, в три раза больше чем в типичном узле Блю Уотерз. С использованием 64 таких узлов, большая модель макета двигателя была рассчитана внутри системы (in-core) за 12 часов.

Когда у нас были результаты расчета при прогоне внутри системы на Крее, мы смогли определить, что обработка входной информации в позиции MPI 0 требовала более 64 ГБ, что составляло объем памяти стандартного узла Блю Уотерз. Тогда NCSA попробовала провести расчет снова, в этот раз прикрепив позицию MPI 0 к одному из 96 узлов с большой памятью системы Блю Уотерз, у которых по 128 ГБ. Это позволило LS-DYNA провести парсинг большой модели макета двигателя и успешно провести решение одной разреженной линейной матрицы до того, как задание было остановлено. На Рисунке 5 показана доступная память на позиции MPI 0 в зависимости от времени. Минимум наблюдается при обработке входных данных, когда используется более 80 ГБ. Узкое место с последовательным символьным разбиением требует второй по величине объем памяти (время 1:10 - 1:30), и с трудом помещается в 64 ГБ.

Только один прогон системы Блю Уотрез с большой моделью макета двигателя был начат до того, как в 2017 году была выпущена новая версия распределения памяти. В апреле 2018 года мы сможем возожновить наше исследование масштабируемости LS-DYNA на Блю Уотерз. Мы удалим из цикла нагрузки (offload) обработку входных данных и распределим символьное разбиение и обработку ограничений. Это должно позволить нам наконец провести расчет большой модели до завершения на Блю Уотерз и начать кампанию по изучению и улучшению работы кода в параллельном режиме У нас мало сомнений в том, что мы сможем наблюдать работу числа арифметических ядер сначала на порядок, а потом и на два порядка больше, чем это было доступно для разработчиков в LSTC, и мы сможем существенно улучшить их параллельное масштабируемость (распараллеливание).

Рисунок 5
Доступная память от времени на позиции MPI 0 при работе с большой большой моделью макета двигателя на 8096 ядрах Блю Уотерз. Всего было 1024 позиций MPI, каждая по 8 потоков, по одной на узел Блю Уотерз

Резюме

Роллс-Ройс поставил перед Креем, LSTC и NCSA задачу продемонстрировать, что аналитические конечно-элементные неявные расчеты больших моделей могут быть осуществлены в приемлемое время при помощи больших компьютерных систем. Модели двигателя, созданные для этого исследования, насколько нам известно, - самые большие неявные модели, когда-либо считавшиеся при помощи LS-DYNA. Это привело к открытию многочисленных проблем в коде LS-DYNA и привычной программной инфраструктуре, которую использует код. Это ускорило переход LS-DYNA на динамическое выделение

памяти и внедрение эвристического вложенного разбиения LS-GPart. После двух лет работы мы смогли провести расчет внутри системы (in-core), с большим числом процессоров и упорядочением вложенного разбиения, что совместно дало 13-кратное сокращение общего времени расчета. Сейчас мы находимся на стадии, когда мы можем начать исследование поведения LS-DYNA при масштабировании, идентифицировать и преодолеть узкие места работы кода. Это должно сделать код LS-DYNA более подходящим для использования на еще бОльших моделях и бОльших компьютерных системах.

Литература

[Tinney 1967] W. F. Tinney and J. W. Walker, Direct solution of sparse network equations by optimally ordered triangular factorization, Proceedings of IEEE, 55 (1967), pp. 1801-1809.

Liu 1985] J. W. H. Liu, Modification of the minimum degree algorithm by multiple elimination, ACM Transactions on Mathematical Software, 11 (1985), pp. 141-153.

[George 1973] J. A. George, Nested dissection of a regular finite element mesh, SIAM Journal on Numerical Analysis, 10 (1973), pp. 345-363.

[Karypis 1995] G. Karypis and V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, Proceedings of the 1995 International Conference on Parallel Processing, 1995, pp. 113-122

[Ashcraft 2016] C. Ashcraft and F.-H. Rouet, A global, distributed ordering library, SIAM Workshop on Scientific Computing, 2016.

[Koric 2016] Koric S. and Gupta A. Sparse Matrix Factorization in the Implicit Finite Element Method on Petascale Architecture. Computer Methods in Applied Mechanics and Engineering, 2016 v.32:281-292