Уфа : УГАТУ, 2011 Вестник УГАТУ Т. 15, № 5 (45). С. 120–129

УДК 004.413+004.94+519.683

В. А. Гасилов, Г. А. Багдасаров, А. С. Болдарев, С. В. Дьяченко, Е. Л. Карташева, О. Г. Ольховская, С. Н. Болдырев, И. В. Гасилова, В. А. Шмыров

СОВРЕМЕННЫЕ МЕТОДЫ РАЗРАБОТКИ ПРОГРАММ ДЛЯ 3D-МОДЕЛИРОВАНИЯ ЗАДАЧ ПЛАЗМОДИНАМИКИ (ПЛАЗМЕННОЙ МУЛЬТИФИЗИКИ)

Представлен опыт использования современных технологий коллективной разработки прикладных программ, накопленный при создании в Институте прикладной математики им. М. В. Келдыша РАН комплекса программ MARPLE, предметной областью которого являются течения ионизированного газа (плазмы), взаимодействующего с магнитным полем. Комплекс MARPLE сконструирован как программное обеспечение для высокопроизводительных параллельных вычислительных систем. Дано общее описание инфраструктуры комплекса MARPLE. Комплекс основан на модели радиационной плазмодинамики, описывающей совокупность нелинейных физических процессов (плазменная «мультифизика»). Рассмотрено моделирование сжатия токонесущей плазменной оболочки собственным магнитным полем. *MARPLE*; *магнитная гидродинамика*; *технологии программирования*; *Z-пинч*

ВВЕДЕНИЕ

Компьютерное моделирование – один из важнейших методов анализа и оптимизации в тех случаях, когда исследования выполняются на основе комплексных физических моделей, называемых моделями «мультифизики». В вычислительной физике плазмы модели такого рода востребованы в весьма высокой степени. Появление высокопроизводительной параллельной техники открывает возможность создания численных кодов, которые могут быть использованы для прогноза результатов экспериментов с высокотемпературной плазмой, выполняемых в целях фундаментальных исследований, управляемого термоядерного синтеза и развития наукоемких технологий.

Темпы исследований в современной физике плазмы очень высокие. Для сопровождения экспериментов необходимо создание программного обеспечения (ПО), которое можно быстро адаптировать к изменившимся условиям и которое содержит средства, упрощающие работу с комплексными моделями, алгоритмами и данными различной структуры. Область применения «параллельного» кода должна быть широкой, чтобы сделать его востребованным для многих исследователей и тем самым оправдать затраты на его создание. Поэтому качество современного ПО для научных исследований должно быть на уровне «индустриальных» кодов, так же как и поддержка труда разработчиков и сопровождение программ. Сложные исследовательские коды, как правило, создаются путем коллективной разработки. Перечисленные требования заставляют обратить особое внимание на архитектуру кода и технологии программирования.

В статье описывается процесс разработки прикладного программного комплекса (кода) MARPLE [1], созданного к настоящему времени в ИПМ им. М. В. Келдыша РАН. Код предназначен для трехмерного (3D) моделирования процессов в импульсной высокотемпературной плазме. Он представляет собой совокупность солверов систем уравнений базовых физико-математических моделей и вычислительной среды (инфраструктуры) для проведения расчетов начально-краевых задач математической физики на массивно-параллельных системах с распределенной памятью.

В основу кода MARPLE положена модель плазмы С. И. Брагинского [2], включающая трехмерное описание гидродинамики плазмы в магнитном поле. Энергетический баланс плазмы учитывает эффект «двухтемпературности», т. е. энергетическую неравновесность электронов и ионов. Учтен ряд важных эффектов, необходимых для описания высокочастот-

Контактная информация: gazizov@mail.rb.ru

Работа выполнена при поддержке программы 3-ОМН РАН, грантов РФФИ 09-01-12109-офи-м, 09-02-01532-а, 10-02-00063-а, 10-02-00449-а, а также Исследовательского Центра Грама (Франция). Расчеты выполнены на компьютерах СКИФ МГУ и МВС-100К МСЦ РАН.

Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии 2011».

ных процессов (эффект Холла, возникновение термоэлектродвижущих сил и т. д.). Модель дополнена описанием лучистого обмена энергией, играющего важную роль в высокотемпературной плазме [3].

В предметную область кода MARPLE в первую очередь входят задачи, связанные с сильноточными импульсными разрядами [4].

Для их решения основная система уравнений может быть дополнена моделью протяженного по времени плазмообразования [5], электротехническим уравнением для электрического тока во внешней электрической цепи (в типовых задачах импульсной энергетики полная цепь включает генератор, подающие линии и разрядную камеру с плазмой) и расчетно-экспериментальными базами данных в форме функциональных или табличных зависимостей (нами использовались данные, полученные с помощью кода TERMOS [6], и широкодиапазонные уравнения состояния вещества [7]).

Реализация этих моделей и организация вычислительных экспериментов осуществляется посредством современных численных методов. Моделирование сложной геометрии экспериментальных и промышленных установок, а также пространственно-разномасштабных физических процессов, выполняется на расчетных сетках нерегулярной структуры, в том числе блочных. Допустимы сетки, содержащие разнородные (тетраэдры, пирамиды, гексаэдры и треугольные призмы) и криволинейные элементы. Для описания таких сеток используется аппарат топологических комплексов [8]. Разработанные и реализованные в MARPLE структуры данных предназначены для генерации трехмерных расчетных сеток, построения на них разностных схем и поддержки распределенных вычислений.

Разработаны оригинальные методики для численного решения уравнений в частных производных на таких сетках. Фундаментальным требованием к технике аппроксимации является консервативность результирующей компьютерной модели. Для бездиссипативной подсистемы полной системы уравнений МГД-модели использовано семейство полностью явных схем, основанных на реконструкциях решения TVD (ограничение полной вариации) [9] повышенной точности с коррекцией потоков. Для подсистемы, описывающей диссипативные процессы, применяются неявные конечно-элементные схемы на основе метода Галеркина с разрывными базисными функциями [10]. Построенные схемы используют средние значения величин по ячейкам вместо узловых значений величин, что позволяет отказаться от рассмотрения трехмерных контрольных объемов сложной формы вокруг узлов сетки; в качестве ячеек консервативности выступают непосредственно ячейки расчетной сетки.

Для поддержки работы с разнообразными физическими моделями, в том числе для обеспечения возможности их модификаций, используется универсальный принцип суммарной аппроксимации, выражающийся в организации вычислений путем последовательного учета физических процессов. Данный принцип делает возможным построение алгоритма продвижения по времени, допускающего независимый расчет всех уравнений, включенных в основную систему. Более того, алгоритм допускает использование на одном временном шаге явных (для гиперболической подсистемы) и неявных (для параболической/эллиптической подсистем) аппроксимаций по времени. Для повышения порядка аппроксимации по времени используется двухступенчатая схема «предиктор-корректор».

Для выполнения расчетов на массивно-параллельных системах с распределенной памятью применяется геометрический параллелизм, основанный на декомпозиции расчетной области (разбиении ее на вычислительные подобласти, каждая из которых обрабатывается отдельным процессорным ядром системы).

В настоящее время код находится в стадии тестирования и пробной эксплуатации.

1. ТЕХНОЛОГИИ РАЗРАБОТКИ КОДА MARPLE

В разработке любой программы можно выделить следующие этапы:

- анализ требований,
- проектирование,
- реализация,
- тестирование,
- выпуск и поддержка.

Анализ требований к программе является самым важным этапом разработки. Здесь закладывается фундамент будущей программы. Должны быть учтены не только текущие требования к программе, но возможность ее модификации в будущем, от чего зависит ее жизненный цикл. Главным результатом этого этапа является список согласованных требований к программе – спецификация.

Проектирование представляет собой формализацию результатов анализа, при которой требования к программе формулируются на языке предметной области в виде алгоритмов (проце-

дурное проектирование) или диаграмм взаимодействующих друг с другом объектов, являющихся экземплярами соответствующих классов, в свою очередь образующих иерархию (объектно-ориентированное проектирование). На этом этапе выбирается язык разработки, вырабатываются правила программирования и оформления кода. Подробно о проектировании и о выбранном нами стиле проектирования рассказано в пп. 2.2.

Реализация заключается в переносе формально записанных требований к программе (в виде алгоритмов или объектов / классов) на какой-либо язык программирования. В отличие от предыдущих этапов, которые выполняют один-два человека (менеджер, главный программист), реализация осуществляется большим количеством разработчиков.

Координацию действий отдельных разработчиков в нашем проекте мы осуществляли с помощью известных, но малоиспользуемых в научной среде инструментов коллективной разработки, таких как система контроля версий, системы управления заданиями и ошибками, средства общения разработчиков (форум, списки рассылки, wiki). Интегрированные системы управления разработкой предоставляют единый интерфейс для перечисленных средств. Средствам коллективной разработки посвящен пп. 2.3.

Тестирование является, наверное, самым сложным этапом разработки. В общем случае оно включает верификацию и валидацию. Верификация разработанного ПО – это формальная проверка работоспособности программы на различных входных данных, часто не имеющих физического смысла. Этот этап очень важен для проверки базовых модулей, фундамента программы. Валидация ПО является специфической процедурой как относительно типа программы, так и начальных требований, и подробно обсуждается на этапе анализа требований. Валидация исследовательских кодов заключается в проведении тестовых расчетов и сравнении полученного решения с известными данными – аналитическими оценками, численными решениями других авторов, результатами экспериментов и наблюдений.

Валидация MARPLE и ему подобных исследовательских кодов весьма важна, во-первых, поскольку они предназначены для предсказания результов еще не состоявшегося эксперимента, во-вторых, ввиду того, что многие модели физики плазмы сами находятся в стадии развития, в том числе и на основе вычислительных экспериментов.

Тестирование программных комплексов выполняется пошагово: тестирование компонент, интеграция, тестирование комплекса. Для каждой компоненты и комплекса необходимо выполнить и верификацию, и валидацию.

Поддержка предлагает обратную связь с пользователями программы: составление документации, обучение, исправление ошибок, улучшение. Мы считаем, что для исследовательских кодов наиболее важным является написание качественной документации для кода в целом, для применяемых алгоритмов и схем, а также для программной реализации. Инструменты для составления и поддержки документации обсуждаются в пп. 2.3.

1.1. Модели разработки

Перечисленные этапы разработки в том или ином виде присущи процессу создания любого ПО, от простейшего калькулятора до операционной системы, разница проявляется лишь во времени, затрачиваемом на каждый этап, и в последовательности этапов.

Нами была выбрана итерационная модель разработки как наиболее рациональная для создания исследовательского кода. Она использует в качестве контролирующего механизма обратную связь, а не планирование, и предназначена для задач с меняющимися или неполными требованиями. В итерационных моделях последовательность этапов разработки повторяется несколько раз: каждый следующий цикл уточняет, совершенствует и расширяет уже имеющийся программный продукт. Это позволяет не только выделить действительно важные требования к продукту, но и изменять и / или исправлять архитектуру программы.

Водопадная модель, подразумевающая однократное последовательное выполнение этапов разработки, в нашем случае неприменима, потому что требования к коду могут меняться на протяжении жизненного цикла программы. Эта модель разработки ПО пригодна для создания систем с фиксированным набором требований.

Техника экстремального программирования, одна из самых известных модификаций итерационной модели разработки, в которой для ускорения итерационного процесса используются такие приемы, как разработка через тестирование, парное программирование, рефакторинг, коллективное владение кодом и другие, также для нас не подходит, поскольку невозможно организовать разработку через тестирование ввиду сложности и слабой формализованности физических тестов. В то же время идеи коллективного владения кодом (каждый разработчик

может вносить изменения в любой участок кода) и рефакторинг (рецензирование кода для улучшения структуры и устранения избыточности) не противоречат итерационной модели и успешно используются в нашей работе.

1.2. Объектно-ориентированное проектирование

Основным и универсальным принципом борьбы со сложностью является принцип «разделяй и властвуй». Разбить большую сложную задачу на ряд более мелких подзадач можно двумя путями: алгоритмическое разбиение, основанное на порядке выполнения процедур, и объектное, когда выделяются объекты и субъекты некоторых действий. Исходя из специфики решения исследовательских задач в определенных сложных пространственных областях, мы выбрали объектно-ориентированный (ОО) подход. Многообразие геометрических данных наиболее естественным образом описывается на языке объектов.

Кроме того, ОО-проектирование предоставляет удобные средства организации кода для обеспечения многовариантности вычислительных моделей и методов. Таким образом, в нашем проекте, наряду с геометрическими объектами используются объекты вычислительные – солверы, аппроксимации, граничные условия и др.

Объектной декомпозиции задачи соответствуют два первых шага разработки программной системы – ОО анализ и проектирование. В основе ОО-проектирования лежит представление о том, что программную систему необходимо проектировать как совокупность взаимодействующих друг с другом объектов, рассматривая каждый объект как экземпляр определенного класса, причем классы образуют иерархию. Для наглядного представления и проектирования взаимодействия объектов и иерархий классов используется унифицированный язык моделирования (Unified Modeling Language – UML). Подробно структура классов MARPLE представлена в пп. 3.1.

Объектно-ориентированное программирование. Для создания программ по объектной модели необходима соответствующая поддержка со стороны языка разработки. Для того, чтобы язык программирования мог считаться объектно-ориентированным, необходимо выполнение следующих условий [11]:

- поддержка объектов, т. е. абстракции данных, имеющих интерфейс в виде именованных операций и собственные данные, с ограничением доступа к ним;
- объекты должны принадлежать к соответствующим типам (классам);
- типы (классы) могут наследовать атрибуты супертипов (суперклассов).

Для проекта MARPLE наиболее подходящей технологией нам представляется ОО программирование на языке C++. Наш опыт использования языка C++ показал, что аппарат производных классов и виртуальных функций является весьма эффективным средством для решения прикладных вычислительных задач с использованием техники нерегулярных расчетных сеток.

Адаптируемая к конкретным условиям архитектура ПО, базирующаяся на программировании средствами C++, предоставляет возможность быстрого обновления кода и делает его удобочитаемым. Разрабатываемое ПО должно быть совместимо с различными операционными системами (MS Windows, GNU/Linux, BSD, Mac OS), работать на различных платформах (персональные компьютеры, мощные рабочие станции, ЭВМ коллективного пользования – включая параллельный процессинг), для которых имеются компиляторы C++ стандарта 2003 года. Развитое прикладное ПО на языке C и C++ совместимо с перечисленными операционными системами и платформами.

1.3. Коллективная разработка

Стандарты программирования. Одним из основных приемов при коллективной разработке является выработка стандартов программирования и единого стиля оформления кода. Следование этим двум стандартам позволило достичь таких важных целей, как переносимость и коллективное владение кодом. Коллективное владение кодом означает, что любой разработчик может легко разобраться и при необходимости исправить код, написанный другими. Облегчается включение в команду новых разработчиков и освоение ими наработанного материала. Переносимость видится нам еще более важной целью, особенно для исследовательских кодов, целевой платформой которых являются высокопроизводительные кластеры с заранее неизвестной архитектурой и установленным программным обеспечением.

Система сборки. Описание проекта в общем случае представляет собой список исходных файлов и опций, необходимых компилятору и линковщику для сборки исполняемого файла. Для крупного проекта таких опций может потребоваться не один десяток (оптимизи-

рующие опции, подключаемые библиотеки и другие). При коллективном создании ПО разработчиками могут использоваться различные аппаратные и программные платформы, а также инструменты для сборки. Все это порождает огромное разнообразие в необходимых опциях и даже в самих компиляторах и линковщиках. Поддержка проектных файлов для всех используемых конфигураций в актуальном состоянии не представляется возможным. Для решения этой проблемы существуют кросс-платформенные системы сборки, такие как GNU Autotools, Scons/waf, qmake, CMake и другие, которые предоставляют унифицированное описание проекта в том или ином виде.

Стандартной системой сборки проекта MARPLE была выбрана система CMake. Файл проекта CMake представляет собой простой текстовый файл с инструкциями на специальном достаточно простом языке макросов. CMake поставляется с большим количеством поисковых макросов, сильно упрощающих запись зависимостей проекта от внешних библиотек и компонентов. Имеется графический интерфейс, предоставляющий наглядный доступ ко всевозможным опциям проекта.

CMake не занимается непосредственно сборкой, а лишь создает файлы управления сборкой для различных платформ и сред разработки. Таким образом, разработчики могут использовать привычные им средства (IDE, компиляторы и другие) для работы с проектом независимо друг от друга.

Система управления версиями. Системы управления версиями (VCS – Version Control System) предназначены для ведения истории развития электронных документов. Они позволяют хранить несколько версий одного и того же документа, легко получать доступ к другим версиям и находить отличия между версиями, определять авторство сделанных изменений. VCS отслеживают конфликты, возникающие при изменении одного файла несколькими разработчиками, и предлагают средства их решения. Поддержка ветвления и слияния процесса разработки открывает еще больше возможностей как для разработчиков, так и для пользователей. Разработчики могут создавать ветки кода для экспериментирования – например, для реализации новой возможности или рефакторинга кода – не мешая другим разработчикам.

В качестве основной VCS для исходных кодов проекта MARPLE была выбрана система Subversion. Это современная и при этом стабильная система, обладающая многими полезными свойствами (кросс-платформенность, атомарность транзакций, поддержка свойств файлов и др.) и подходящая для разработки программ небольшим коллективом.

Репозиторий проекта MARPLE включает общепринятую иерархию каталогов для хранения исходных файлов и файлов проекта:

- trunk – основная ветка проекта (общая версия, основная для разработчиков);
- branches – альтернативные ветви проекта (специальные рабочие версии, которые планируется в дальнейшем объединить с основной веткой проекта);
- tags – неизменяемые, официально выпущенные и протестированные сборки MARPLE, которые можно считать релиз-версиями.

Документация. Для MARPLE, как и для большинства исследовательских кодов, характерно большое число настроечных параметров, заданных во внешних файлах или внутри программного кода. Наличие подробного руководства для запуска, с примерами, является необходимым условием для распространения кода как за пределами команды разработчиков, так и для внутреннего пользования.

Описание различных программных решений является необходимым условием для коллективного владения кодом разработчиками. Это помогает при отладке и локализации ошибок, рефакторинге, ускоряет обучение новых разработчиков. Документирование программных решений часто оформляется в самом коде в виде комментариев. При следовании определенным правилам оформления комментариев становится возможным автоматически извлечь из них информацию и сформировать полное описание программного кода в распространенных форматах (HTML, RTF и другие).

Мы пользуемся кросс-платформенной системой Doxygen, которая является общепринятым программным средством для автоматического создания документации по исходному коду. Doxygen создает документацию на основе комментариев к исходному коду программы, оформленных в специальном стиле, и может быть настроен на извлечение и сохранение структуры программы (графы зависимостей программных объектов, диаграммы классов и исходных кодов с гиперссылками). Doxygen поддерживает множество языков программирования (C, C++, Fortran, Python, Java и др.) и имеет встроенную поддержку создания документации в формате HTML, LaTeX, man, RTF и XML. Опыт разработки MARPLE свидетельствует в пользу формата HTML – работа в нем значи-

тельно проще для объемной документации с множеством перекрестных ссылок.

Система управления проектом. Система управления проектом – это комплекс программ, включающий приложения для планирования задач, составления расписания, контроля цены и управления бюджетом, распределения ресурсов, совместной работы, общения, быстрого управления, документирования и администрирования системы. Различают несколько типов таких систем: desktop- и web-ориентированные, персональные, одно- и многопользовательские, а также интегрированные. На рынке ПО представлено огромное число реализаций каждого типа систем управления проектом, как коммерческих, так и имеющихся в свободном доступе. Выбор конкретного инструмента зависит от многих факторов: численности и распределения коллектива разработчиков, требований к разрабатываемой системе, имеющихся ресурсов и других.

В качестве системы управления проектом MARPLE была выбрана web-ориентированная система trac, предназначенная для разработки ПО небольшим коллективом. К важным для нас преимуществам этой системы можно отнести простоту установки, конфигурирования, администрирования, гибкость, наличие большого числа специальных плагинов, интеграцию с любыми современными VCS, простоту освоения и работы.

Для организации работы выбранных нами средств (Subversion, trac) был создан специальный GNU/Linux-сервер, предоставляющий доступ к хранилищу исходных текстов (репозиторию) MARPLE и системе trac. Наиболее используемыми нашим коллективом возможностями trac стали: интеграция с Subversion, позволяющая быстро просматривать историю кода и конкретные изменения, совместное редактирование документов (wiki) для оформления отчетов и управление заданиями/ошибками (билетами). Последнее для нас оказалось чрезвычайно эффективным средством, помогая разрешению проблем, требующих участия нескольких разработчиков.

2. АРХИТЕКТУРА КОДА MARPLE

Проект разработки кода для проведения параллельных расчетов начально-краевых задач математической физики в трехмерной постановке был начат в ИММ РАН и продолжается по настоящее время в ИПМ им. М. В. Келдыша РАН [1]. Разработка кода изначально велась с применением обсуждавшихся в п. 1 технологий разработки.

2.1. Проектирование кода MARPLE

Основные идеи, заложенные при создании MARPLE (как абстрактной вычислительной среды, так и солверных компонентов), это:

- реализация большого числа сервисных функций (ввод-вывод данных, операции с расчетными сетками, поддержка параллельных вычислений, динамическая работа с вычислительными объектами) на уровне вычислительной среды, что позволяет создавать физические солверы с минимальными затратами ресурсов и обеспечивает унификацию их интерфейсов;
- автоматическое управление динамическим созданием и уничтожением вычислительных объектов разных типов, таких как солверы уровней вычислительного домена, физической подобласти и элементарные солверы, аппроксимаций, граничных условий.

В соответствии с этими идеями и принципами объектно-ориентированного проектирования предметная область была разбита на модули: аппроксимация, работа с сетками, солверы, граничные условия и уравнения состояний. Иерархия классов в модулях, а также отношения между ними представлены на рис. 1 (показаны только существенные классы и отношения, рамками выделены модули):

- главные солверы (MainSolver) – совокупность физических солверов, выполняют операции верхнего уровня с физическими подобластями, реализуют схему суммарной аппроксимации и схемы аппроксимации по времени;
- физические солверы (PhysicalDomainSolver) – совокупность элементарных солверов, решают систему уравнений в одной физической подобласти;
- элементарные солверы (ElementalSolver), отвечают за решения какого-либо уравнения или системы уравнений;
- элементарные граничные условия (ElementalSolverBC), свои для каждого элементарного солвера, учитывают специфику решаемых элементарным солвером уравнений на границе;
- сложные граничные условия (PhysicalDomainBC) уровня физической подобласти, используются только в том случае, когда не удается разложить заданное граничное условие на множество элементарных граничных условий, находящихся ниже в иерархии (ElementalSolverBC);

- аппроксимации (Approximation), предоставляют унифицированный доступ ко всем необходимым для работы элементарных солверов аппроксимациям;
- свойства вещества (MatterProperties), предоставляют унифицированный доступ к уравнениям состояния для каждой физической подобласти.

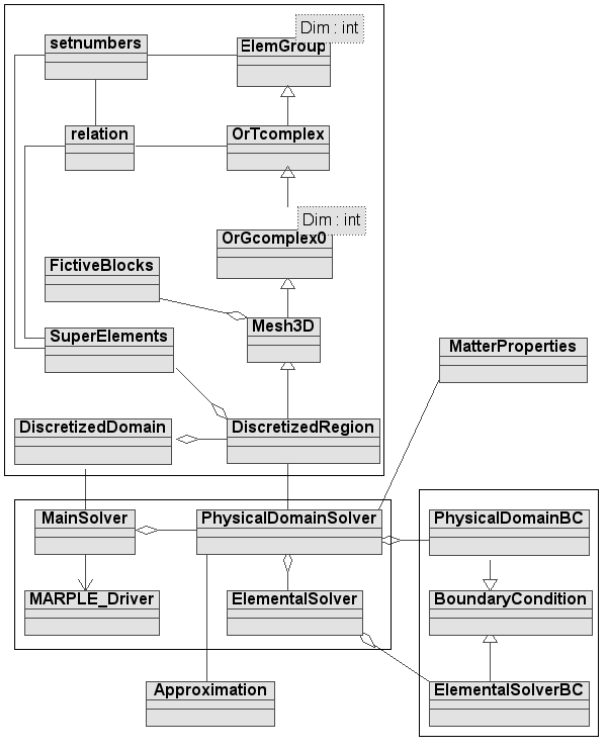

Рис. 1. Диаграмма классов UML: архитектура MARPLE

Выделение аппроксимаций в отдельный модуль с унифицированным интерфейсом позволяет различным вычислительным объектам (солверам и самим аппроксимациям) пользоваться одними и теми же объектами для схожих операций. Совместно используемая база аппроксимаций MARPLE обеспечивает автоматизированное управление коллективным доступом к вычислительным объектам, предотвращение излишних затрат памяти и ее возможных утечек, возможность динамических замен вычислительных объектов, упрощение и унификацию сборки кода.

В коде используются смешанные сетки нерегулярной структуры. Многоуровневая иерархия классов дикретной модели, представленная на рис. 1, основана на аппарате топологических комплексов [8] для описания таких сеток. Обобщенная реализация классов позволяет применять их для описания дискретизаций областей любой размерности.

Поддержка геометрического параллелизма встроена в дискретную модель в виде блоков фиктивных элементов (FictiveBlocks и Mesh3D на рис. 1) и средств эффективного обновления расчетных величин в этих блоках (актуализации). Фиктивный блок каждого вычислительного домена содержит сеточные элементы из соседних доменов со специальной маркировкой, позволяющей вычислительным объектам отличать «свои» элементы от соседних. Фиктивные блоки также применяются нами для описания различных симметрий в расчетных областях, например, периодических граничных условий.

Наличие нескольких физических подобластей в расчетной области также отражено в иерархии классов дискретной модели (DiscretizedRegion и DiscretizedDomain на рис. 1). Объект класса DiscretizedRegion описывает дискретизацию одной физической подобласти на одном вычислительном домене. Для постановки граничных условий между различными физическими подобластями, а также для выделения границ расчетной области, применяются так называемые «суперэлементы», представляющие собой списки сеточных элементов, принадлежащих границам.

Для обеспечения баланса загрузки вычислительных узлов на каждом из них размещаются фрагменты всех имеющихся в расчетной области физических подобластей. Для этого расчетная сетка с M физических подобластей при расчете на N вычислительных узлах разбивается на $M \cdot N$ кусков (каждая физическая подобласть разбивается на N кусков). Список дискретизаций физических подобластей содержится в объекте класса DiscretizedDomain и обрабатывается в главном солвере (объекте класса MainSolver).

2.2. Сторонние разработки

Использование открытых и широко распространенных сторонних программ (библиотек) при разработке крупных исследовательских кодов избавляет разработчиков от необходимости тратить время на программирование и отладку вспомогательных систем, применяемых для решения конкретных задач, отделяемых от основного кода. В проекте MARPLE используется несколько сторонних библиотек для осуществления таких функций, как декомпозиция расчетной области, параллельное решение систем алгебраических уравнений, компрессия данных и их визуализация.

Для более компактного хранения внешних данных (файлы с сеткой, выходные файлы для визуализации, файлы для сохранения / восста-

новления расчета) нами используется библиотека zlib. Библиотека написана на языке C, и работа с ней из кода C++ недостаточно прозрачна и подвержена ошибкам. Имеющиеся в свободном доступе классы-обертки (zipstream, gzstream) предоставляют стандартный для C++ способ ввода/вывода данных через потоки.

В рамках проекта MARPLE разработан специальный набор утилит для подготовки к расчетам распределенных сеточных данных. Для эффективной декомпозиции сеточных графов в нем используется один из наиболее известных и широко применяемых инструментов для решения такого рода задач – библиотека ParMETIS.

Для решения в распределенном режиме больших систем линейных алгебраических уравнений, получаемых при построении неявных схем, мы используем библиотеку Aztec. В ней реализованы параллельные итерационные алгоритмы, основанные на методах подпространства Крылова, применяемые при решении больших линейных систем, которые могут быть плохо обусловлены или получены при моделировании нестационарных процессов. В свою очередь эта библиотека использует функции фактически стандартных пакетов BLAS и LAPACK, которые зачастую предустановлены в вычислительных системах производителем и оптимизированы под конкретную архитектуру и набор компиляторов. Разработанные нами классы обеспечивают работу с распределенными разреженными матрицами и упрощают взаимодействие вычислительных модулей (солверов) с функциями библиотеки Aztec, которая также написана на языке C.

В качестве основы для реализации наших методов и алгоритмов на многопроцессорных системах используется широко распространенная среда параллельного программирования MPI. Набор функций, необходимых для работы кода MARPLE, также поддерживается специальными классами-обертками, которые помимо обеспечения интерфейса, более свойственного языку C++, значительно упрощают взаимодействие разработчиков остальных модулей со средой MPI. При проектировании и написании любого модуля должна обеспечиваться принципиально корректная работа кода в режиме распределенных данных и его выполнения на большой многопроцесорной системе. При этом многоуровневые иерархии солверных и сеточных классов практически полностью скрывают от разработчика вычислительного модуля технические детали параллелизма.

3. МОДЕЛИРОВАНИЕ СЖАТИЯ ПРОВОЛОЧНОЙ СБОРКИ ТОКОВЫМ ИМПУЛЬСОМ

Эксперименты с самосжимающимися разрядами (пинчами) проводятся на сильноточных генераторах в ряде лабораторий России, США, Англии и Франции. Их целью является получение мощного импульса рентгеновского излучения (мощностью от нескольких Твт до сотен ТВт при длительности от нескольких нс до десятков нс). Такие импульсы могут применяться для нагрева термоядерной мишени в схемах инерционного УТС, в нанотехнологиях и для фундаментальных исследований в области лабораторной астрофизики, изучения экстремальных состояний вещества и др. Источником плазмы в этих экспериментах может служить проволочная сборка – конструкция из тонких металлических проволок или полимерных нитей, натянутых между электродами. Высота и диаметр сборки имеют сантиметровые размеры, диаметр проволок – несколько микрон. На электроды подается мощный импульс тока (амплитуда от 1 до 20 МА, время нарастания от 100 нс до 1 мкс). Под действием этого импульса проволочки нагреваются, испаряются и переходят в плазменное состояние. Возникающее азимутальное магнитное поле сжимает плазму к оси сборки, где в момент максимального сжатия происходит быстрый переход кинетической энергии в энергию излучения.

В данном примере использовалась постановка задачи, приведенная в [4]. Проволочная сборка – «цилиндр», составленный из 240 алюминиевых проволок (диаметр 10.4 мкм, высота 50 мм). Диаметр сборки 140 мм, диаметр разрядной камеры 150 мм. Расчет выполнен при заданном токе генератора (амплитуда 6 МА, время нарастания 700 нс). Уравнения состояния, транспортные коэффициенты и оптические свойства использованы в форме таблиц [6].

Для расчета генерации плазмы при взрыве проволок сильноточным импульсом использовалась нестационарная модель плазмообразования [5]. Проволочная сборка в расчете заменяется эффективной «плазмообразующей поверхностью», а неоднородность потока плазмы моделируется стохастическим возмущением (до 5%) темпа появления плазмы.

Расчетная сетка – блочная, из смешанных элементов (шестигранники и треугольные призмы), построена в цилиндрическом секторе 60º, состоит из приблизительно 4 млн ячеек (рис. 2). Средний линейный размер ячейки $h_\varphi \sim 600$–400 мкм, $h_r \sim 375$ мкм и $h_z = 400$ мкм. На вертикальных плоских гранях сектора были заданы пе-

риодические граничные условия, что эквивалентно полномасштабному 3D-моделированию в цилиндре. Такой подход представляется весьма перспективным для экономичных расчетов на подобных сетках.

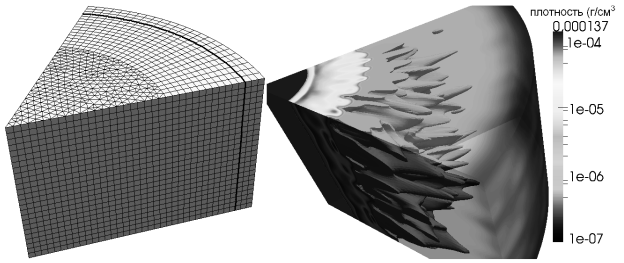

Рис. 2. Слева: пример блочной сетки в цилиндрическом секторе (7 тыс. ячеек), справа: развитие неустойчивости на периферии плазменной оболочки (4 млн ячеек)

Гидродинамическая неустойчивость, возникающая при сжатии пинчей, может оказать заметное влияние на достижимые параметры. Исследование таких неустойчивостей – одна из важных задач 3D-моделирования, так как неустойчивость плазмы, ускоряемой электродинамически, является существенно трехмерным эффектом.

В данном примере на периферии пинча можно видеть развитую стадию неустойчивости плазмы. Существенные возмущения формы оболочки в виде отклонений от цилиндрически симметричного движения вещества могут замедлить образование финальной структуры, снизить температуру и плотность пинчевой плазмы. С другой стороны, турбулизация течения плазмы вызывает повышенную диссипацию электрической энергии токового импульса, вследствие чего растет температура вещества. Таким образом, расчет показывает результат конкуренции различных, притом нелинейных, процессов. Анализ расчетных данных показал, что в результате моделирования были получены физически адекватные результаты, совпадающие с экспериментальными данными по таким показателям, как мощность излучения, время сжатия, размер, плотность и температура пинча. Хорошее соответствие с результатами экспериментов показали также расчеты пинчей при других начальных и краевых условиях. Полученные при апробации кода данные свидетельствуют в пользу того, что построенная 3D–модель Z-пинча является состоятельной для данного класса задач и может быть использована для предсказания результатов будущих экспериментов.

ЗАКЛЮЧЕНИЕ

Рассказано о современных технологиях разработки крупных программных комплексов, представлены различные инструментальные средства, используемые разработчиками для повышения эффективности и скорости создания программ.

Описанные методы программирования и инструментальные средства использовались для разработки программного кода MARPLE, предназначенного для проведения параллельных расчетов прикладных начально-краевых задач. Использование смешанных сеток нерегулярной структуры обеспечивает возможность вычислений в расчетных областях нетривиальной геометрии.

Разработанный код MARPLE применен для моделирования сжатия токовым импульсом алюминиевой проволочной сборки – одной из типовых задач импульсной энергетики. Полученные результаты качественно верно воспроизводят динамику плазмы и основные особенности сжатия проволочных сборок.

План развития кода предусматривает включение в него новых моделей импульсной плазмы, а также оптимизацию процессов вычислений и обработки данных. В перспективе предусматривается подготовка кода к использованию на вычислительных системах сверхвысокой (тера- и эксафлопной) производительности, которые в предстоящее десятилетие будут использоваться в научных исследованиях.

СПИСОК ЛИТЕРАТУРЫ

1. Object-Oriented Programming and Parallel Computing in RMHD Simulations / V. Gasilov [et al.] // IOS Press: Advances in Parallel Computing. 2008. Vol. 15.

2. **Брагинский С. И.** Явления переноса в плазме // Вопросы теории плазмы / под ред. М. А. Леонтовича). М.: Атомиздат, 1963. Вып. 1, С. 183–272.

3. **Chandresekhar S.** Radiative transfer. Oxford, 1951.

4. **Фортов В. Е.** Экстремальные состояния вещества. М.: ФИЗМАТЛИТ, 2009.

5. Динамика гетерогенного лайнера с затянутым плазмообразованием / В. В. Александров [и др.] // Физика плазмы. 2001. Т. 27, № 2.

6. **Никифоров А. Ф., Новиков В. Г., Уваров В. Б.** Квантово-статистические модели высокотемпературной плазмы. М.: ФИЗМАТЛИТ, 2000.

7. Метастабильные состояния жидкого металла при электрическом взрыве / И. В. Ломоносов [и др.] // ТВТ. 2001. Т. 39, №5. С. 728–742.

8. An Implicit Complexes Framework for Heterogeneous Objects Modelling, in Heterogeneous Objects Modelling and Applications / E. Kartasheva [et al.] // Lecture Notes in Computer Science. 2008. Vol. 4889. P. 1–41.

9. **Куликовский А. Г., Погорелов Н. В., Семенов А. Ю.** Математические вопросы численного решения гиперболических систем уравнений. М.: ФИЗМАТЛИТ, 2001.

10. **Cockburn B., Shu C. W.** The local DG method for time-dependent convection-diffusion systems // SIAM Journal of Numerical Analysis. 1998. 35 (6). P. 2440–2463.

11. **Буч Г.** Объектно-ориентированный анализ и проектирование с примерами приложений на C++. СПб.: Невский диалект, 2001.

ОБ АВТОРАХ

Гасилов Владимир Анатольевич, проф., зав. отд. ИПМ им. М. В. Келдыша РАН. Дипл. инженер-физик (МФТИ, 1975). Д-р физ.-мат. наук по матем. моделированию (ИММ РАН, 1992). Иссл. в обл. вычислительн. математики и матем. моделирования в механике сплошных сред.

Багдасаров Геннадий Алексеевич, мл. науч. сотр. ИПМ им. М. В. Келдыша РАН. Дипл. инженер по прикладной математике и информатике (МИФИ (ГУ), 2008). Иссл. в обл. вычислительн. математики и матем. моделирования в механике сплошных сред.

Болдарев Алексей Сергеевич, зав. сект. ИПМ им. М. В. Келдыша РАН. Канд. физ.-мат. наук по матем. моделированию, числ. методам и комплексам программ (ИММ РАН, 1999). Иссл. в обл. вычислительн. математики и матем. моделирования в механике сплошных сред.

Дьяченко Сергей Валерьевич, науч. сотр. ИПМ им. М. В. Келдыша РАН (г. Москва), доц. ИАТЭ НИЯУ МИФИ (г. Обнинск). Дипл. инженер по прикладн. математике (МИФИ (ГУ), 2003). Канд. физ.-матем. наук по матем. моделированию, числ. методам и комплексам программ (ИММ РАН, 2006). Иссл. в обл. вычислительн. математики и матем. моделирования в механике сплошных сред.

Карташева Елена Леонидовна, зав. сект. ИПМ им. М. В. Келдыша РАН. Дипл. инженер-математик по прикладн. математике (МИФИ, 1986). Канд. физ.-мат. наук по матем. моделированию, числ. методам и комплексам программ (ИММ РАН, 1994). Иссл. в обл. геометрическ. моделирования, вычислительн. геометрии, генерации расчетных сеток и подготовки данных численного моделирования.

Ольховская Ольга Гургеновна, ст. науч. сотр. ИПМ им. М. В. Келдыша РАН. Дипл. инженер-системотехник (МИФИ, 1984). Канд. физ.-матем. наук по матем. моделир. (ИММ РАН, 1993). Иссл. в обл. вычислительн. математики, матем. моделирования в механике сплошных сред.

Болдырев Сергей Николаевич, канд. физ-мат. наук по матем. моделированию, числ. методам и комплексам программ (ИММ РАН, 2001). Иссл. в обл. параллельных вычислений на системах с распр. памятью.

Гасилова Ирина Владимировна, асп. ИПМ им. М. В. Келдыша РАН. Дипл. инженер по автоматизирован. системам обработки информации и управления (МИФИ, 2009). Иссл. в обл. вычислительн. математики и матем. моделирования в механике сплошных сред.

Шмыров Валерий Александрович, асп. МГТУ «Станкин». Дипл. магистр техники и технологии по информатике и вычислительн. технике (МГТУ «Станкин», 2008). Иссл. в обл. вычислительн. математики и матем. моделирования в механике сплошных сред.