УДК 681.3.069, 681.324

ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ПРОГРАММНЫХ КОМПЛЕКСОВ ДЛЯ МОДЕЛИРОВАНИЯ СЛОЖНЫХ СИСТЕМ

С. В. Ковальчук,
младший научный сотрудник
С. В. Иванов,
младший научный сотрудник
И. И. Колыхматов,
стажер-исследователь
А. В. Бухановский,
доктор техн. наук, профессор
Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Рассматриваются особенности проектирования высокопроизводительных программных комплексов для моделирования сложных систем на примере задачи моделирования экстремальных явлений в атмосфере и океане. Обсуждается подход к формализации структуры комплекса, сочетающий в себе специфику концепций SOA и REST, в соответствии с задачей отображения компонентов сложной системы на параллельную вычислительную архитектуру. Предложен способ оценки архитектуры на основе анализа параметрических моделей параллельной производительности.

Введение

Высокоуровневое проектирование (или разработка архитектуры ПО — программного обеспечения) ставит своей целью формализацию и обоснование внутренней структуры ПО, которая наилучшим образом удовлетворяет проектным требованиям при заданном наборе ограничений в области работоспособности, безопасности, безотказности, защищенности [1].

Эволюция способов формализации структуры ПО определяется, в первую очередь, потребностью снижения сложности процессов разработки, тестирования и поддержки программных продуктов на фоне увеличения общей сложности решаемых задач. Это последовательно порождает структурный [2], объектно-ориентированный [3] и компонентно-ориентированный [4] подходы в инженерии ПО. Дальнейшая тенденция функциональной изоляции компонентов отражается в архитектурах на базе SOA (Service Oriented Architecture) [5], REST (Representational State Transfer) [6], AOA (Aspect Oriented Architecture) [7] и др. Эти подходы достаточно успешно используются при разработке коммерческих приложений и часто могут определять весь технологический процесс создания ПО [8].

При разработке наукоемкого ПО (scientific computing software engineering) необходимо учитывать, что приоритеты проектных требований, их структура, процесс проектирования, создания, проверки, внедрения, поддержки и использования обладают рядом индивидуальных особенностей [9]. Во многих случаях важным критерием работоспособности ПО является производительность (productivity), характеризующая получение результата вычислений с приемлемой точностью за заданное время. Она обеспечивается за счет применения технологий высокопроизводительных вычислений в многопроцессорных и (или) распределенных средах.

Использование традиционных компонентно-ориентированных подходов к проектированию распределенных программных систем (на основе DCOM [10], CORBA [11], Enterprise JavaBeans [12] и пр.) допустимо не во всех случаях, поскольку они не ориентированы непосредственно на достижение наибольшей производительности. Несмотря на то что для наукоемкого ПО предлагаются специфические проблемно-ориентированные подхо-

ды, в частности CCA (Common Component Architecture) [13], в общем случае архитектура таких систем определяется особенностями решаемой задачи, спецификой предметной области и характеристиками параллельной вычислительной системы. В данной статье обсуждаются особенности проектирования наукоемкого ПО для моделирования сложных систем на примере высокопроизводительного программного комплекса моделирования экстремальных явлений в атмосфере и океане, разрабатываемого в рамках проекта ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2012 годы».

Специфика компьютерного моделирования сложных систем

Моделирование сложных систем (complex system simulation) является одним из приоритетных направлений развития наукоемкого ПО, порождающим новый этап развития «электронной» науки (e-Science) [14]. Под сложной системой [15] понимается система, которая:

а) состоит из очень большого количества элементов;

б) допускает «дальние» связи между элементами;

в) обладает многомасштабной (пространственно-временной) изменчивостью. Как следствие, поведение системы описывается целым комплексом взаимодействующих моделей (в общем случае основанных на различном математическом аппарате и соответствующих различным областям знания) и требует существенных вычислительных затрат.

Характерным примером сложной системы является система «океан-атмосфера», рассматриваемая в диапазонах мелкомасштабной, синоптической, сезонной, межгодовой изменчивости (временные масштабы от секунд до десятилетий и пространственные — от метров до тысяч километров). Она характеризуется гидрометеорологическими полями атмосферного давления, скорости ветра, температуры воздуха, морского волнения, температуры и солености, течений и уровня моря. Необходимость исследования такой системы связана с задачей проектирования судов и сооружений на шельфе, что требует оценок экстремальных гидрометеорологических явлений как сочетаний характеристик, возможных один раз в T лет. Величина T определяется инженерными требованиями к конкретному объекту или сооружению и составляет от 10 до 10 000 лет. Экстремальные характеристики обычно не обеспечены данными наблюдений и, как следствие, должны определяться на основе компьютерного моделирования [16].

Процедура моделирования соответствует функциональной схеме, представленной на рис. 1. На первом этапе осуществляется подготовка метеорологических данных «А» (атмосферного давления и ветра) за климатический интервал (от 30 лет) для выполнения непрерывных расчетов полей вол-

■ ***Рис. 1.*** *Функциональная схема программного комплекса моделирования экстремальных гидрометеорологических явлений*

нения, течений и уровня моря по гидродинамическим моделям «Б» с учетом верификации результатов расчетов «В» и их калибровки по данным измерений «Г». После этого выполняется параметризация «Д» и последующая статистическая обработка однородных выборок больших объемов данных «Е». Целью обработки является идентификация семейства стохастических моделей «Ж», по которым методом Монте-Карло (с использованием зависимых испытаний) воспроизводится ансамбль многомасштабной изменчивости гидрометеорологических полей. На основании этих данных выполняется оценивание экстремальных (ненаблюдаемых) явлений «И», например сочетаний скоростей ветра, высот волн и скоростей течений в шторме, возможном один раз в 100 и более лет, по которым рассчитывается степень риска для инженерных объектов «К». Подробнее применяемые методы и технологии описаны в работах [16, 17].

Ресурсоемкость расчетов существенно зависит от расчетной акватории и характеристик задачи (например, периода повторяемости T); в общем случае она составляет порядка нескольких петафлоп. Например, для Каспийского моря выполнение таких расчетов на одном процессоре AMD Opteron 275 занимает 446 суток [18]. Поэтому каждый из функциональных компонентов «А»—«К» должен предусматривать параллельное выполнение.

Поскольку экстенсивный подход на уровне автоматического распараллеливания программных кодов [19] далеко не всегда эффективен, разработка высокопроизводительного ПО моделирования сложных систем требует формализации параллельной структуры компонентов на этапе проектирования. В то же время применение традиционного для параллельных вычислений подхода на основе PCAM-концепции [20] затруднено тем, что механизм моделирования сложных систем включает в себя иерархическую систему моделей, связанных между собой, основанных на различном математическом аппарате и имеющих различную ресурсоемкость. С точки зрения проектирования структуры ПО, это отражается в том, что:

- функциональные компоненты могут иметь ограничения по использованию — функциональные, платформенные, лицензионные и пр.;
- функциональные компоненты могут быть написаны на различных языках, могут использовать разнообразные системы ввода-вывода;
- структура исходных кодов компонентов (в случае их доступности и возможности внесения изменений) может быть трудномодифицируемой вследствие индивидуальных особенностей разработки;
- компоненты используют различные интерфейсы параллельного программирования и ориентированы в общем случае на разные вычислительные архитектуры.

Например, на этапе «Б» для гидродинамического моделирования волнения применяется свободно-распространяемая модель SWAN [21], которая реализована в стандарте языка Fortran 90 и распараллеливается с использованием интерфейсов OpenMP и MPI [22]. Эта модель признана мировым сообществом; любая ее модификация (содержательная или программная) требует проведения дополнительной сертификации. С другой стороны, на этапе «И» для расчета экстремальных характеристик гидрометеорологических процессов применяется авторская модель BOLIVAR. Несмотря на то что эта модель рекомендована Всемирной метеорологической организацией [23], коллектив разработчиков имеет права на модификацию ее кодов, в то время как модификация самого метода потребует дополнительных согласований.

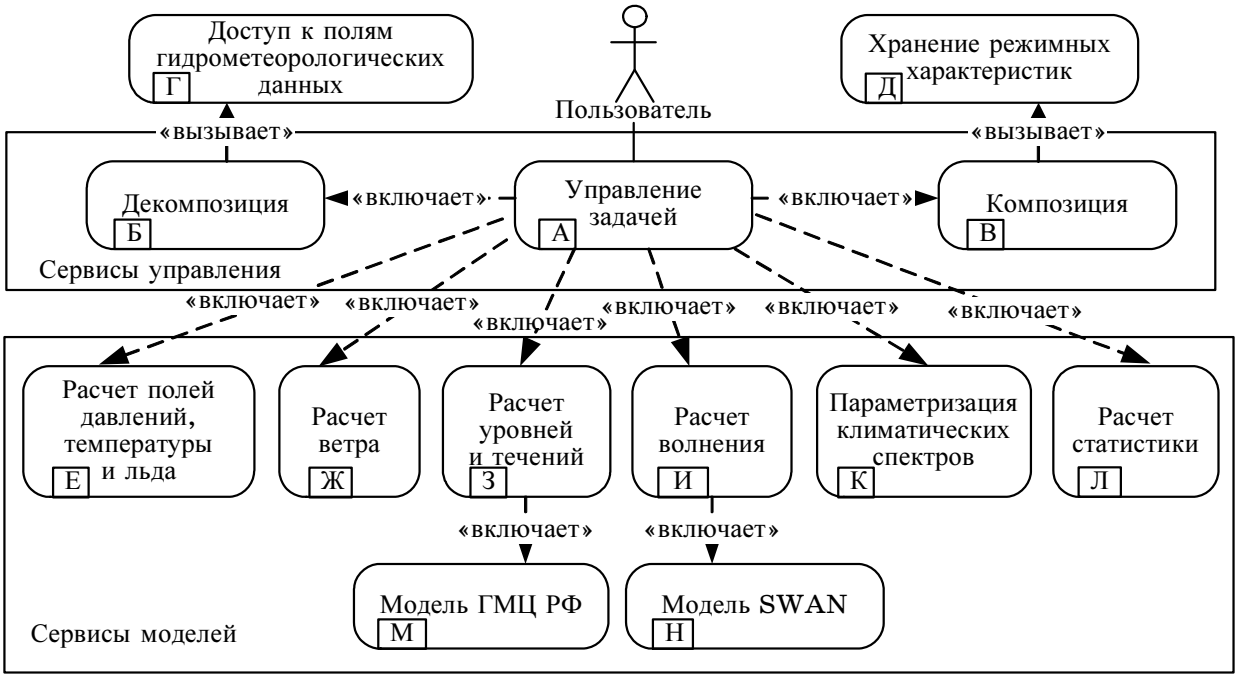
Специфика разработки архитектуры высокопроизводительного программного комплекса состоит в том, что он должен соответствовать архитектуре вычислительной системы. Ниже рассматривается решение, ориентированное на перспективный суперкомпьютерный вычислительный комплекс Росгидромета РФ, состоящий из двух кластеров: кластера SGI Altix 4700 с пиковой производительностью 11 ТФлопс и кластера SGI ICE 8200 с пиковой производительностью 16 ТФлопс.

Разработка программной архитектуры

Классический подход к разработке архитектуры программной системы предусматривает операции структурирования системы, описания свойств ее компонентов и установления связей между ними. Для ПО моделирования сложных систем структурные компоненты соответствуют различным функциональным компонентам, например «А»—«К» на рис. 1. Потому в данном случае принципиальной проблемой является организация взаимодействия (как между компонентами, так и внутри них) таким образом, чтобы обеспечивалось наиболее эффективное использование ресурсов вычислительной системы.

Сервисно-ориентированная архитектура. В общем случае сложная система описывается иерархией моделей, соответствующих различным диапазонам изменчивости и функциональным подсистемам. При этом диапазоны изменчивости связаны между собой параметрически, т. е. выходные данные, полученные по одной модели, являются управляющими данными для модели более низкого уровня. Как следствие, соответствующее ПО может быть интерпретировано в рамках сервисно-ориентированной архитектуры (SOA) [5]. Это позволяет добиться максимального уровня изоляции частей программного комплекса, тем самым делая более строгой его структурированность и облегчая процесс разработки и тестирования. На рис. 2 представлена общая структура взаимодействия сервисов, соответствующая функциональной схеме на рис. 1. Для нее характерна следующая типизация сервисов [24].

- Управляющие сервисы (композиции, декомпозиции, управления задачей).



■ ***Рис. 2.*** *Сервисно-ориентированная архитектура программного комплекса моделирования экстремальных гидрометеорологических явлений в форме диаграммы вариантов использования*

• Сервисы доступа к данным (входная гидрометеорологическая информация, результаты расчетов).

• Вычислительные сервисы (модели подготовки метеорологической информации, гидродинамические и стохастические модели, статистическая обработка и расчет экстремумов).

В общем случае функциональные компоненты, указанные на рис. 1, могут быть представлены в виде совокупности сервисов, исполняющихся на отдельных узлах вычислительной системы и обеспечивающих наиболее полное использование вычислительных ресурсов. Следует отметить, что часть сервисов присутствует в единственном экземпляре, обеспечивая централизованное управление комплексом моделей в целом, в то время как другая часть исполняется на каждом из узлов вычислительной системы.

Вычислительные модули представляют собой сервисы доступа к используемым моделям. Взаимодействие моделей организуется управляющим модулем вычислительной системы, на который ложится ответственность за определение потоков данных между сервисами моделей. В то же время потоки данных в рамках сервисов также обладают довольно сложной структурой, обусловленной фиксированным интерфейсом модели. Таким образом, каждый из сервисов моделей обязан предоставить интерфейс для унифицированного обращения с моделью, организуя потоки данных таким образом, чтобы эффективно использовать вычислительные ресурсы системы.

Использование SOA позволяет путем формализации интерфейса взаимодействия с сервисом упростить процедуру разработки управляющих модулей. При совмещении данного решения с применением паттерна проектирования «Адаптер» [25] становится возможной легкая модификация и замена одних моделей другими с аналогичной функциональностью. Изоляция сервисов и четкое описание интерфейса позволяют упростить схему взаимодействия, не учитывая при этом особенностей внутренней реализации каждого из разрабатываемых компонентов. В рамках решаемой задачи для реализации сервисов была использована среда .NET Framework, реализующая технологию Windows Communication Foundation (WCF), что в общем случае дает возможность строить кросс-платформенные сервисы различных видов.

Предоставление доступа к данным. Моделирование сложных систем связано с оперированием большими объемами информации. Потому в рамках данной задачи применяется технология REST, ориентированная на обеспечение эффективного представления ресурсов [6, 26]. Доступ к ресурсам осуществляется через фиксированный интерфейс, что обеспечивает унификацию ресурсов и упрощает взаимодействие с пользователем. Совместное использование технологий SOA и REST позволяет добиться организации унифицированного доступа к обрабатываемым данным. На рис. 2 часть рассмотренных сервисов («Г», «Д») отвечает за предоставление доступа к большим объемам данных, необходимых для работы программного комплекса.

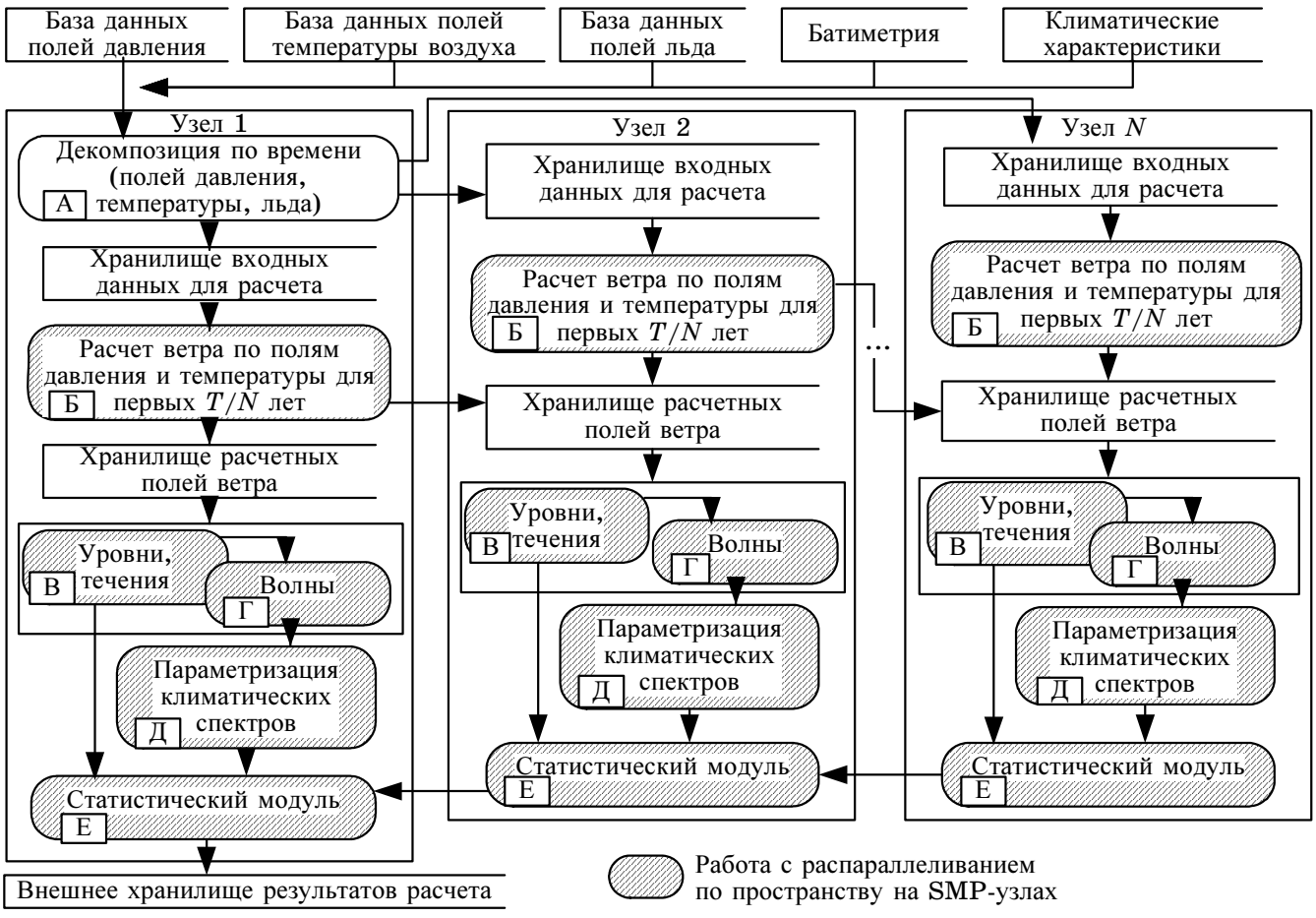
В процессе организации взаимодействия системы с разделяемой памятью возникает необходи-

мость решения задачи эффективной коммуникации. При использовании среды исполнения .NET Framework возможна передача данных при помощи механизма сериализации/десериализации данных. Этот механизм позволяет осуществить преобразование объектов, доступных программному коду, к форме, удобной для передачи посредством механизма потоков. Это дает возможность оптимизировать процесс передачи данных по сети за счет сжатия, синхронизации и дополнительного контроля передачи данных путем как декларативного, так и императивного управления процессом сериализации/десериализации [27].

Несмотря на то что для коммерческого ПО традиционным способом хранения данных являются реляционные базы данных, для представления расчетной гидрометеорологической информации эффективнее использовать файловые форматы GRIB и NetCDF. Однако каждому из расчетных файлов сопоставляются метаданные (характеристики сетки, условия расчета и пр.), которые допустимо хранить в реляционной базе данных.

Отображение на параллельную вычислительную архитектуру. В процессе разработки программного комплекса необходимо учитывать особенности аппаратной архитектуры и среды исполнения, что позволяет добиться эффективной работы программного комплекса в целом.

Задача отображения архитектуры программного комплекса на параллельную вычислительную архитектуру определяет соотношение четырех уровней взаимодействия между компонентами системы. *На аппаратном уровне* взаимодействие происходит в рамках системы, состоящей из ряда соединенных коммуникационной средой вычислительных узлов, на каждом из которых установлен один или более процессоров (ядер), имеющих доступ к общей для них памяти. *Системный уровень взаимодействия* оперирует объектами операционной системы, процессами, потоками, нитями в пределах одного вычислительного узла и взаимодействиями на уровне сетевых коммуникаций при синхронизации различных узлов. *Программный уровень* предполагает применение библиотечных средств, предназначенных для коммуникации, синхронизации, балансировки и других задач организации параллельных вычислений. К средствам, традиционно относимым к реализации параллельных вычислений, можно причислить MPI (для систем с разделенной памятью)



■ ***Рис. 3.*** *Схема отображения программной архитектуры на параллельную вычислительную архитектуру*

и OpenMP (для систем с общей памятью). Средства, позволяющие осуществлять межпроцессное взаимодействие, обеспечивают возможность коммуникации между отдельными процессами в рамках локального узла. Возможны различные способы реализации IPC (Inter-Process Communication) как с использованием механизма потоков, каналов, очереди сообщений MSMQ (Microsoft Message Queuing) и пр., так и более высокоуровневые реализации вплоть до возможности работы .NET Remoting посредством IPC. Наконец, *логический уровень* представляет собой сам вычислительный алгоритм, который выполняется с использованием средств нижних уровней, обеспечивая оптимальную загрузку вычислительных ресурсов системы.

В рамках данного программного комплекса реализуется иерархическая схема распараллеливания. Она включает в себя компоненты трех категорий.

- Центральный управляющий модуль, предоставляющий внешним подсистемам сервис для осуществления вычислений и использующий сервисы узлов для непосредственного контроля процесса работы.
- Исполняющие модули, производящие вычисления на узлах и работающие с каждым из вычислительных компонентов.
- Адаптированные модели, реализующие параллельный вариант расчетных алгоритмов для систем с общей памятью.

На рис. 3 представлена схема отображения. Центральный управляющий модуль (ввода-вывода) выполняет предварительные операции с данными. Затем происходит распределение массива метеорологических характеристик (давление, температура воздуха) и льда объемом T лет между p вычислительными узлами. После этого выполняется процедура усвоения, а также по полям давления и температуры рассчитывается ветер, «Б». Эта процедура на каждом узле распараллеливается по пространству. Производится репликация данных между смежными узлами, что необходимо для последующей сшивки результатов расчетов по гидродинамическим моделям «В» — «Г». На каждом узле совместно (на каждом шаге) выполняется гидродинамическое моделирование волнения, течений и уровня моря, также распараллеленное по пространственной области в рамках общей памяти узла. Затем результаты расчетов передаются в модуль параметризации «Д». Полученные в итоге поля параметров передаются в статистический модуль, выполняется агломерация результатов. Статистические модули «Е» на каждом узле также допускают многоуровневое распараллеливание — по ансамблю, по индексирующей переменной, по таксонам, на основе принципа перемешивания [28].

Использование сервисно-ориентированного подхода позволяет гибко реализовывать различные схемы параллельных вычислений, допуская как одновременную обработку независимых задач, так и распараллеливание внутри каждой задачи, в зависимости от специфики вычислительной архитектуры. Это дает возможность организовывать иерархические схемы параллельных вычислений на системах гибридной (или NUMA, Non-Uniform Memory Architecture) архитектуры.

Оценка архитектуры

Оценка программной архитектуры устанавливает степень соответствия предложенной архитектуры поставленным требованиям. Для многих классов бизнес-приложений оценка архитектуры строится на неформализованных подходах (например, оценка сценариев). Однако с точки зрения проектирования высокопроизводительных программных систем предоставляется возможность количественной оценки основной проектной характеристики — параллельной производительности системы (времени работы, ускорения, эффективности и пр.).

Для оценки архитектуры на этапе проектирования необходимым становится прототипирование и моделирование разрабатываемого ПО для прогнозирования особенностей его использования и формализации требований. Ниже представлены оценки, полученные на примере моделирования климатической системы Каспийского моря на мини-суперкомпьютере TForge-Mini.

В рамках схемы на рис. 3 можно приближенно описать время выполнения произвольного сервиса на p вычислителях с общей памятью (внутри одного узла) в форме

$$T_{calc}(p,\alpha,\gamma) = T_0\left[\left[\alpha(p-1)+\frac{1}{p}\right]\cdot\gamma+(1-\gamma)\right], \quad (1)$$

где α — параметр, характеризующий рост накладных расходов с увеличением количества вычислителей, а γ — доля параллельных вычислений. Например, применительно к акватории Каспийского моря процедуре подготовки метеорологической информации соответствует $\gamma = 0{,}96$, расчету волн — $\gamma = 0{,}95$, расчету течений и уровня моря — $\gamma = 0{,}59$, параметризации климатических спектров — $\gamma = 0{,}98$.

Общее время коммуникаций между узлами определим как сумму времени рассылки и сбора данных, а также времени на передачу перекрытий между независимыми блоками данных:

$$T_{comm} = t_w\left[\left(3L_0\tau_0+\Pi\mu L\right)+\Pi L\vartheta+ \right.$$
$$\left.+\left(\eta\xi L+\nu(\xi L)^2\right)(\Pi-1)\right], \quad (2)$$

где τ_0 — число полей метеорологических (входных) данных по времени; L_0, L — число узлов расчетной сетки по пространству для метеорологических (входных) и океанографических (выходных) данных; Π — количество узлов; η, ν, ξ, μ — коэффициенты, характеризующие кратность использования данных в моделях статистики; ϑ — коли-

чество реплицируемых полей между узлами. Число полей океанографических (выходных) данных по времени τ не влияет на объем собираемых с удаленных узлов данных, так как модели статистической обработки результатов применяются непосредственно к тем океанографическим полям, которые были рассчитаны на данном узле. Величина t_w соответствует времени передачи единицы информации по коммуникационной сети. Предполагается, что время работы T_0 каждой задачи (расчета ветра, волн, течений, параметризации спектров и расчета статистики) определяется только производительностью вычислителя (например, ядром CPU) и объемом обрабатываемых данных. Например, для одного ядра AMD Opteron 275 можно записать:

$$T_{0(wind)} = \left[1{,}2\cdot 10^{-5}\cdot\frac{L_0\tau_0}{\Pi}+60\cdot 10^{-5}\cdot\frac{L\tau}{\Pi}\right];$$

$$T_{0(waves)} = \left[8{,}25\cdot 10^{-4}\cdot L\cdot\left(\frac{\tau}{\Pi}+\vartheta_{wave}\right)\right];$$

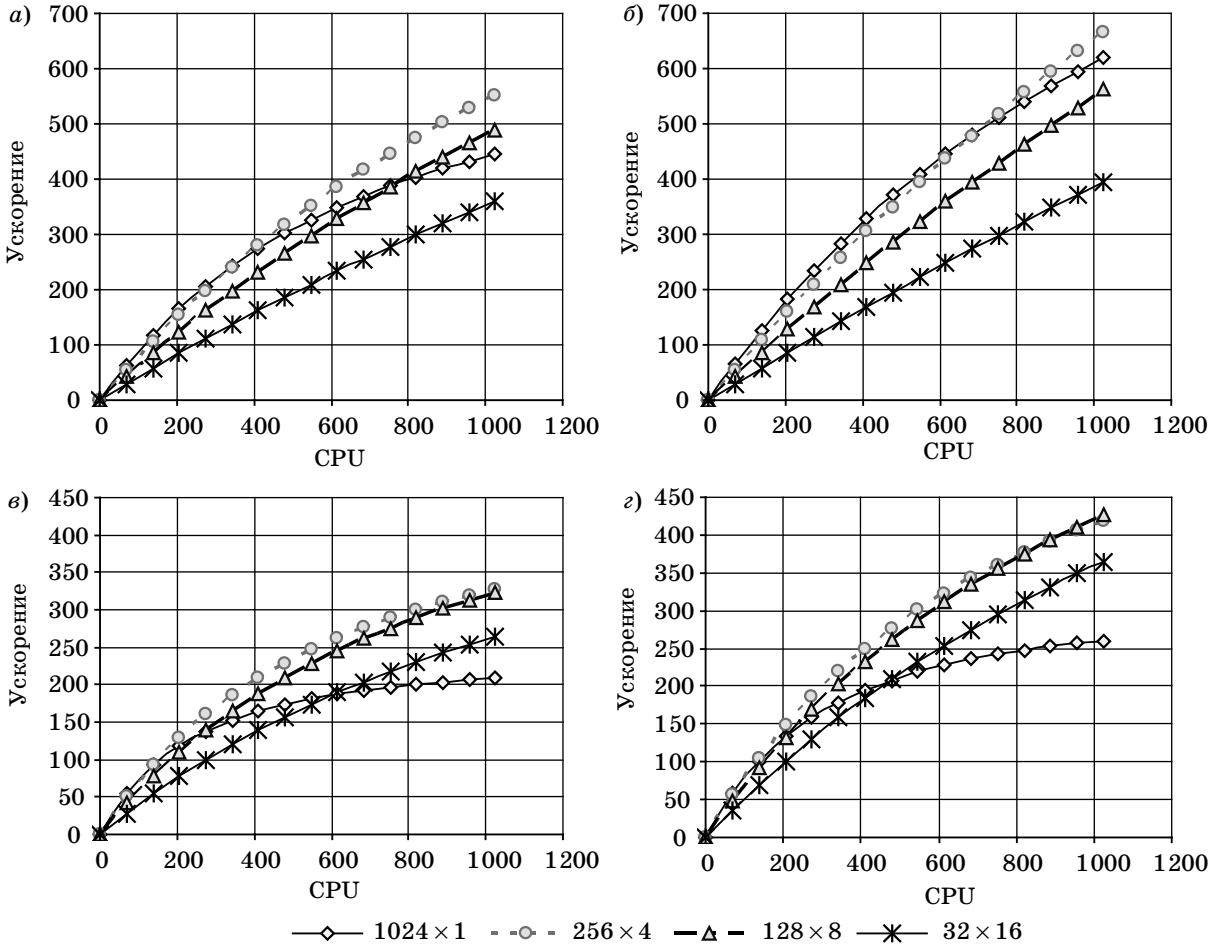
$$T_{0(current\_level)} = \left[21{,}6\cdot 10^{-5}\cdot L\cdot\left(\frac{\tau}{\Pi}+\vartheta_{level}\right)\right];$$

$$T_{0(spectra)} = \left[\xi\cdot 0{,}16\cdot L\cdot\left(\frac{\tau}{\Pi}\right)\right];$$

$$T_{0(statistics)} = \frac{5\cdot 10^{-5}\cdot\xi\cdot\left(L\tau+0{,}07\cdot L^2+0{,}15\cdot L\tau\ln(\tau)\right)}{\text{Пр}}. \quad (3)$$

В выражениях (3) численные коэффициенты перед объемами выполняемых операций имеют смысл времени расчета элементарной операции (в пересчете на одну точку исходных или модельных данных, в секундах); объем обрабатываемых данных делится пропорционально между П узлами. Учитывая, что общее время вычислений является суммой времен, определяемых (2) и (3), этого достаточно, чтобы оценить параллельное ускорение как функцию от параметров.

На рис. 4 показано влияние архитектуры программной системы на общее ускорение вычислений. При небольшом количестве узлов оптимальным является распараллеливание по времени



■ ***Рис. 4.*** *Параллельное ускорение в зависимости от конфигурации вычислительных узлов, длины входного массива и размера перекрытия независимых блоков данных: а — 15 лет, перекрытие 1 месяц; б — 30 лет, перекрытие 1 месяц; в — 5 лет, перекрытие 3 месяца; г — 5 лет, перекрытие 1 месяц*

с использованием отдельных процессоров узлов с общей памятью как независимых вычислителей. Однако с увеличением числа процессоров тенденция изменяется и многопроцессорные узлы становятся более предпочтительными. С другой стороны, чем большее число лет T обрабатывается, тем большее число процессоров необходимо для получения эффекта от применения многопроцессорных узлов. При этом очевидно, что чем большее число процессоров используется в узлах, тем дальше будет находиться точка заката кривой ускорения. Стоит отметить, что с увеличением размера входного массива (число моделируемых лет, рис. 4, *а, б*) в результате увеличения гранулярности общее ускорение также возрастает.

Рис. 4, *в, г* иллюстрирует влияние перекрытия независимых блоков данных (что является обязательным при распараллеливании по времени) на производительность алгоритма. При уменьшении величины перекрытия с трех месяцев до одного ускорение в среднем вырастает примерно в полтора раза, а эффективность от использования многопроцессорных узлов сдвигается в зону большего числа узлов и меньшего числа процессоров на узле. Эта характеристика может меняться в зависимости от специфики акватории (океанские процессы обладают большим «временем жизни», чем процессы в закрытых морях), а также от особенностей моделируемого процесса (синоптический интервал атмосферных процессов и волнения составляет несколько суток, а эволюция вихрей водных масс может длиться несколько месяцев).

Результаты проведенного анализа могут рассматриваться как формализация знаний по управлению параллельными процессами в целях выбора оптимальной по производительности конфигурации для заданной задачи в рамках интеллектуального сервиса управления вычислительными модулями (см. рис. 2, «А»).

Выводы

Существующие стандарты по проектированию ПО (IEEE 1016–1998 и IEEE 1471–2000) носят лишь рекомендательный характер в области определения понятий, а не их содержания. Потому в данной работе на примере высокопроизводительного комплекса моделирования экстремальных гидрометеорологических явлений рассматриваются основные аспекты проектирования наукоемкого ПО для моделирования сложных систем. К ним относятся:

- использование сервисно-ориентированной архитектуры, отображающей взаимодействие управляющих и вычислительных компонентов, соответствующих моделям в различных диапазонах изменчивости;
- оптимизация механизмов оперирования большими объемами данных за счет применения технологии REST и комбинированного способа хранения расчетной информации;
- применение иерархической схемы отображения архитектуры программной системы на параллельную архитектуру вычислительной системы на аппаратном, системном, программном и логическом уровнях;
- использование аппарата параметрических моделей производительности, настроенных на основе прототипов, для оценки степени соответствия архитектуры программного комплекса проектным требованиям и оптимизации производительности программного комплекса за счет динамического выбора схемы отображения.

Предлагаемые решения не ограничены только данной предметной областью. В частности, авторы считают целесообразным использовать их для моделирования биологической сложной системы — глобальной популяции вируса иммунодефицита человека [29, 30].

Литература

1. **Соммервилл И.** Инженерия программного обеспечения. 6-е изд.: Пер. с англ. М.: Издательский дом «Вильямс», 2002. 624 с.
2. **Вирт Н.** Алгоритмы + структуры данных = программы. М.: Мир, 1985. 406 с.
3. **Буч Г.** Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд.: Пер. с англ. М.: «Бином»; СПб.: «Невский диалект», 1999. 560 с.
4. **Szyperski C.** Component Software: Beyond Object-Oriented Programming. Addison-Wesley Professional, 1998. 411 p.
5. **Lublinsky B.** Defining SOA as an architectural style. 9 January 2007. http://www.ibm.com/developerworks/architecture/library/ar-soastyle/
6. **Fielding R. T.** Architectural Styles and the Design of Network-based Software Architectures: Doctoral thesis. University of California, Irvine, 2000. 162 p.
7. **Navasa A., Pérez M. A., Murillo J. M.** Aspect Modeling at Architecture Design // Lecture Notes in Computer Science. 2005. Vol. 3527. P. 41–58.
8. **Якобсон А., Буч Г., Рамбо Дж.** Унифицированный процесс разработки программного обеспечения. СПб.: Питер, 2002. 496 с.
9. **Carver J. C., Kendall R. P., Squires S. E. et al.** Software Development Environments for Scientific and Engineering Software: A Series of Case Studies: Proc. of the 29th International Conf. on Software Engineering. Washington, DC, USA: IEEE Computer Society Press, 2007. P. 550–559.
10. **Sessions R.** COM and DCOM: Microsoft's Vision for Distributed Objects. N. Y., USA: John Wiley & Sons, 1998. 492 p.
11. Object Management Group. OMG's CORBA website. http://www.corba.org
12. **Matena V., Stearns B.** Applying Enterprise JavaBeans: Component-Based Development for the J2EE Platform. Prentice Hall, 2000. 464 p.

13. **Bernholdt D. E., Elwasif W. R., Kohl J. A., Epperly T. G. W.** A Component Architecture for High-Performance Computing: Proc. of the Workshop on Performance Optimization via High-Level Languages and Libraries. N. Y., USA, 22 June 2002.
14. **Sloot P. M. A., Frenkel D., Vorst H. A. Van der et al.** Computational e-Science: Studying complex systems in silico. A National Coordinated Initiative. White Paper, February 2007. http://www.science.uva.nl/research/scs/papers/archive/Sloot2007a.pdf
15. **Boccara N.** Modeling Complex Systems. N. Y.: Springer, 2004. 397 p.
16. Справочные данные по режиму ветра и волнения Балтийского, Северного, Черного, Азовского и Средиземного морей / Под ред. Л. И. Лопатухина, А. В. Бухановского, В. А. Рожкова и др. СПб.: Российский Морской Регистр Судоходства, 2006. 450 с.
17. **Бухановский А. В., Лопатухин Л. И., Иванов С. В.** Подходы, опыт и некоторые результаты исследований волнового климата океанов и морей. I. Постановка задачи и входные данные // Вестник СПбГУ. Сер. 7. 2005. Вып. 3. С. 62–74.
18. **Бухановский А. В., Зильберштейн О. И., Иванов С. В. и др.** Моделирование экстремальных явлений в атмосфере и океане как задача высокопроизводительных вычислений // Параллельные вычислительные технологии (ПаВТ 2008): Тр. Междунар. науч. конф. Санкт-Петербург, 28 января — 1 февраля 2008. Челябинск: ЮУрГУ, 2008. С. 57–68.
19. Compiler Optimization for Scalable PC / Pandle S., Agrawal D. P. (Eds.) // Lecture Notes in Computer Science. 2001. Vol. 1808.
20. **Foster I.** Designing and Building Parallel Programs: Concepts and Tools for Software Engineering. Reading, Massachusetts: Addison-Wesley, 1995. 381 p.
21. **Cazes J. E., Campbell T., Rogers E.** OpenMP Parallel Implementation of SWAN // Navo MSRC Navigator. Fall 2001. P. 13.
22. **Grama A., Gupta A., Karypis G., Kumar G.** Introduction to Parallel Computing. Second Edition. Addison-Wesley, 2003. 856 p.
23. **Lopatoukhin L. J., Rozhkov V. A., Ryabinin V. E. et al.** Estimation of extreme wind wave heights // Reports of World Meteorological Organization. 2000. WMO/TD-No. 1041. 71 p.
24. **Cohen S.** Ontology and Taxonomy of Services in a Service-Oriented Architecture // The Architecture Journal. 2007. Vol. 11. P. 30–35.
25. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.** Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб.: Питер, 2007. 366 с.
26. **Vinoski S.** REST Eye for the SOA Guy // IEEE Internet Computing. 2007. Vol. 11. N 1. P. 82–84.
27. **Obasanjo D.** XML Serialization in the .NET Framework. In Microsoft MSDN Library. 23 January 2003. http://msdn2.microsoft.com/en-us/library/ms950721.aspx
28. **Boukhanovsky A. V., Ivanov S. V.** Stochastic simulation of inhomogeneous metocean fields. Part III: High-performance parallel algorithms // Lecture Notes in Computer Science. 2003. Vol. 2658. P. 234–243.
29. **Иванов С. В., Колыхматов И. И., Бухановский А. В.** Параллельные алгоритмы моделирования комплексных сетей // Параллельные вычислительные технологии (ПаВТ 2008): Тр. Междунар. науч. конф. Санкт-Петербург, 28 января — 1 февраля 2008. Челябинск: ЮУрГУ, 2008. С. 395–402.
30. **Sloot P. M. A., Boukhanovsky A. V., Keulen W. et al.** A GRID-based HIV expert system // Journal of Clinical Monitoring and Computing. 2005. Vol. 19. P. 263–278.