Разработка методов прогнозирования масштабируемости приложений на конфигурации суперкомпьютеров*

К.П. Казьмина, А.С. Антонов

Московский государственный университет имени М.В. Ломоносова

Динамика характеристик исполнения параллельного приложения на различных конфигурациях вычислительной системы не всегда поддается тривиальному описанию. На масштабируемость приложения могут влиять значения входных параметров, особенности реализуемого алгоритма, задействованные аппаратные ресурсы, состояние вычислительной системы и другие факторы. Чтобы корректно прогнозировать масштабируемость с учетом совокупности этих факторов, при построении модели можно задействовать эмпирические данные. В статье рассматриваются различные подходы к прогнозированию масштабируемости приложений и предлагается использование результатов запусков приложения на малых конфигурациях вычислительной системы для предсказания значений характеристик исполнения этого приложения на конфигурациях большего размера. Применимость предложенного подхода проверялась на суперкомпьютерах МГУ имени М.В.Ломоносова «Ломоносов» и «Ломоносов-2».

Ключевые слова: масштабируемость, прогнозирование, регрессия, предсказание производительности.

1. Введение

Параллельные вычисления имеют широкое применение в области прикладных наук и моделирования. Для решения некоторых поставленных задач необходимо использовать мощную вычислительную систему, допускающую высокую степень параллелизма. Архитектура современных суперкомпьютеров включает в себя сотни и тысячи вычислительных узлов. Например, суперкомпьютеры МГУ имени М.В.Ломоносова «Ломоносов» и «Ломоносов-2» имеют соответственно 5104 и 1472 вычислительных узла [1].

Однако, несмотря на высокую производительность ресурсов, доступных для решения вычислительных задач, эти ресурсы могут быть использованы неэффективно. Неэффективность использования ресурсов нежелательна как для пользователей системы, так и для суперкомпьютерного центра. Пользователям приходится дольше ждать размещения своих задач для счета на большом количестве процессоров или завершения работ программ других пользователей, запросивших большое количество процессоров для вычислений. Суперкомпьютеры в этом случае потребляют больше энергии и позволяют выполнить меньшее количество задач пользователей за определенный промежуток времени, чем было бы возможно при более эффективном использовании ресурсов.

Предсказание масштабируемости параллельных программ позволяет не только улучшить эффективность использования вычислительных ресурсов, но также оценить качество самих параллельных приложений, указать пользователю на присутствующие узкие места и охарактеризовать используемую систему. В общем случае динамика характеристик масштабируемости параллельного приложения на реальной вычислительной системе нетривиальна, поэтому чисто аналитический подход к её предсказанию представляется сложным. Зависимость от совокупности влияющих на масштабируемость факторов может быть отражена с учетом эмпирических данных по запускам приложения. Таким образом, надо сначала определить, какие именно данные стоит собирать и в какой форме следует их учитывать.

* Работа выполняется при поддержке РФФИ, грант 16-07-01003. Результаты получены с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова.

Целью нашей работы является построение универсального метода предсказания масштабируемости параллельных приложений, который не подразумевает наличие жестких ограничений на само приложение и на вычислительную систему, на которой происходят запуски этого приложения. Предлагается использование эмпирических данных по запускам без внесения каких-либо изменений в код программ, то есть построение прогноза возможно с использованием набора входных параметров для приложения и соответствующего исполняемого файла.

В данной статье рассматриваются существующие подходы к прогнозированию масштабируемости. Их описание представлено в разделе 2. Далее в разделе 3 обсуждаются разрабатываемые методы предсказания и описываются методы, применимость которых была проверена на суперкомпьютерах «Ломоносов» и «Ломоносов-2» (раздел 4).

2. Подходы к предсказанию масштабируемости приложений

Понятие предсказания масштабируемости параллельных программ трактуется неоднозначно. Если рассматривать широкое определение масштабируемости (как свойства параллельной программы, характеризующего зависимость изменения всей совокупности динамических характеристик работы этой программы от множества параметров её запуска [2]), то чаще всего подразумевается предсказание зависимости времени выполнения параллельной программы от некоторых параметров. В других источниках встречается прогнозирование производительности выполнения программы, её эффективности, производительности за доллар, энергопотребления и его соотношения с производительностью.

Существуют исследования прогнозирования как для сильной масштабируемости, так и для слабой масштабируемости, а также для масштабируемости вширь. При переносе результатов с нескольких конкретных запусков на всё пространство возможных запусков используется экстраполяция результатов запусков с малых частичных конфигураций на случаи больших конфигураций или интерполяция результатов в точках, выбранных из пространства параметров относительно равномерно, на всё это пространство параметров. Предсказания производятся при помощи статистических методов регрессии, нейронных сетей, частичной или полной симуляции исполнения программы и других методов.

Предлагаемые в некоторых статьях решения основываются на разделении времени вычислений и времени коммуникаций, выделении и классификации однородных сегментов исполнения, использовании циклов простоя (stalled cycles), представлении характеристик разных программ как характеристик результатов исполнения бенчмарков, применении методов для выделения из всего множества параметров только тех, которые значительно влияют на результат, и др. Если прогнозируемая величина зависит от нескольких параметров, то авторами рассмотренных работ отмечается, что экстраполяция этих параметров по-отдельности даёт значительно лучший результат, чем экстраполяция некой зависимой объединяющей их величины.

2.1 Линейная регрессия

Использование линейной зависимости для предсказания масштабируемости может предоставить результаты с ошибкой, не превышающей ошибку для других подходов.

В [3] предлагается предсказывать масштабируемость, анализируя результаты выполнения приложения на малых конфигурациях и экстраполируя их на большие конфигурации. Предсказания ведутся в условиях сильной масштабируемости, где это возможно, и слабой масштабируемости в случаях, когда сильная масштабируемость не применима из-за ограничений на объём оперативной памяти процессоров. Считается, что известно, какие из параметров запуска наиболее влияют на время выполнения программы и что вычисления хорошо сбалансированы в системе.

Время выполнения предсказывается для запуска программы на произвольном количестве процессоров p с использованием результатов нескольких запусков той же программы на q процессорах, где $q \in \{2, \ldots, p_0\}$, $p_0 < p$, с различными наборами входных параметров $(x_1, x_2, \ldots, x_n)$. Далее исследуется соотношение параметров и полученного времени выполнения и строится предиктор времени выполнения программы $T = F(x_1, x_2, \ldots, x_n, q)$ на q процессах. Так как модель оценивается с помощью относительной ошибки, время выполнения приближается функ-

цией F в рамках логарифмической шкалы. Предложенная модель имеет следующий вид, линейный в статистическом понимании (далее используется регрессия методом наименьших квадратов):

$$\log_2(T) = \beta_1\log_2(x_1) + \beta_2\log_2(x_2) + \ldots + \beta_n\log_2(x_n) + g(q) + \text{error}$$

В зависимости от приложения, выбирается вид g(q). Предлагается два варианта этого слагаемого: $g(q) = \gamma_0 + \gamma_1\log_2(q)$ и $(q) = \gamma_0 + \gamma_1\log_2(q) + \gamma_2(\log_2(q))^2$ (при этом в статистическом смысле модель всё еще остается линейной).

Авторами рассматривается три подхода к решению поставленной задачи. В первом подходе при регрессии величина T понимается просто как время выполнения программы, все предсказания строятся в рамках таких времён. В остальных двух подходах разделяется время коммуникаций и время вычислений (время вычислений в общем случае при сильной масштабируемости изменяется пропорционально изменению количества процессоров, изменение времени коммуникаций более сложное). Для проведения такого разделения может быть необходимым наличие знаний о коде программы или модификация этого кода. Второй подход основывается на поиске максимального по всем процессорам времени вычислений на одном процессоре и использовании его же времени коммуникаций. Третий подход основывается на определении времени коммуникаций и вычислений на критическом пути (наиболее длинном промежутке выполнения программы, на котором не встречается блокировок). Подходы, в которых время коммуникаций и вычислений разделяется, подразумевают и их раздельную регрессию, причем для каждого из времён существует два способа выбрать вид g(q). Таким образом, для проверки этого подхода на 7 приложениях авторам понадобилось 2 различных вида моделей, и на приложения наложено условие сбалансированности, поэтому подход не представляется полностью универсальным.

В [4] один из подходов также основан на статистической линейной регрессии. Здесь прогнозируется соотношение потребления энергии к производительности для OpenMP-приложений в зависимости от конфигурации на конкретной SMP-системе из двух четырехъядерных процессоров. Предложенные методы предсказания масштабируемости работают на данных, полученных в результате непродолжительного тестового периода исполнения, охватывающего все конфигурации. Далее моделируется функция, отражающая Instructions-Per-Cycle (IPC) для каждой из конфигураций, и, с использованием результатов счетчиков производительности, создается набор параметров, подающихся на вход модулю предсказаний. Для обучения используется линейная регрессия с подбором коэффициентов методом наименьших квадратов, производится предварительный анализ данных. Медианная и средняя относительная ошибка для регрессии, соответственно, 5.6% и 9.4%. Такой подход ориентирован на оптимизацию работы только процессоров, а не суперкомпьютера в целом, однако он также показывает возможности использования для предсказаний линейной регрессии.

Линейная регрессия используется в [5] для предсказания производительности за доллар в зависимости от параметров и размера кластера для параллельного приложения, запущенного на Amazon EC2. Подход, описанный в [3], используется здесь как для предсказания времени выполнения, так и для предсказания стоимости в зависимости от системы оплаты и других параметров.

Также линейная регрессия используется в [6] для предсказания масштабируемости вширь. Авторы представляют подход к предсказанию времени выполнения, основанный на фрактальной теории: приложения, в которых можно выделить превалирующую над остальными (>70% по времени) фазу вычислений, имеющую полиномиальную сложность, имеют высокую степень самоподобия времени выполнения по размеру данных. Линейная регрессия используется для экстраполяции времени вычисления и процентной составляющей этой главенствующей фазы вычислений. В большинстве случаев ошибка предсказаний оказалась меньше 6%. Такой подход также накладывает сильные ограничения на запускаемое приложение и не обладает универсальностью.

В [7] была предпринята попытка с помощью линейной регрессии вычислить производительность параллельного приложения на данном наборе параметров через некоторую линейную функцию от производительностей фиксированного набора бенчмарков. Результаты тестирования подхода показали, что, независимо от того, какой набор бенчмарков выбирался, ошибка

предсказания производительности оказывалась очень большой и составляла 51-72%, и данный подход оказался неуспешным.

2.2 Регрессия с использованием других аналитических функций

Выбор нелинейных функций для описания масштабируемости приложений теоретически позволяет описать более сложную и неоднородную зависимость в сравнении с линейной. Далее описываются подходы, позволяющие осуществить прогнозирование на основе нелинейных аналитических функций.

Готовым и ориентированным на широкое применение решением является ESTIMA[8]. Оно позволяет предсказывать время исполнения приложения, для работы которого достаточно только оперативной памяти, на большом количестве узлов, используя результаты запусков на малом числе узлов (при этом рассматривается возможность осуществления тестовых запусков не на целевой системе). Метод основывается на работе с циклами простоя на узле – тактами непродуктивной работы приложения, во время которых происходит ожидание данных при кэш-промахе, ожидание снятия блокировки для осуществления доступа к ресурсу и др. Производительность приложения может сильно зависеть от циклов простоя, если они занимают значительную по времени часть исполнения в сравнении со временем вычислений, что встречается при работе с разделяемыми ресурсами, то есть в условии вычислительных систем с общей памятью.

Регрессионный анализ для экстраполяции значений циклов простоя, по которым далее вычисляется время исполнения, происходит следующим образом: с помощью нелинейных ядер создается функция, аппроксимирующая результаты запусков на всех тестовых конфигурациях, кроме нескольких наибольших; наибольшие конфигурации используются для уточнения предсказаний посредством минимизации среднеквадратичной ошибки. Экстраполяция ведется отдельно для каждого типа циклов простоя. Ошибки предсказаний в условиях сильной и слабой масштабируемости не превосходят 25%.

В [9] авторы предлагают метод предсказания производительности параллельных приложений, основанный на соотнесении их фаз исполнения с базой данных производительностей набора бенчмарков. Функции зависимостей производительности от размера данных для этих бенчмарков были смоделированы с использованием кубических сплайнов для регрессии, основанной на нескольких тестовых запусках с разными размерами входных данных. При этом процентная составляющая выделенных фаз моделируется с помощью линейной регрессии.

В [10] один из подходов к моделированию производительности параллельных приложений основывается на использовании кусочно-полиномиальной регрессии. Данные, на которых осуществляется анализ, выбираются равномерно случайным образом из всего пространства параметров разными способами. Представленные методы построения прогноза теоретически могут быть применены и для предсказания на основе эмпирических данных по запускам на малых конфигурациях.

Производится предобработка набора данных для осуществления лучшего моделирования производительности. Для того, чтобы выбрать наиболее репрезентативные для моделирования производительности параметры, производится иерархическая кластеризация с корреляцией в качестве метрики. Таким образом можно избавиться от избыточных для построения предсказаний параметров: если несколько параметров сильно коррелируют, они могут быть объединены в один кластер. При небольшом доступном количестве экспериментов уменьшение числа параметров снижает вероятность неправильного подбора регрессионных коэффициентов.

Анализ ассоциаций позволяет качественно оценить соотношения параметров и характеристик производительности, то есть насколько этот параметр важен для оценки, что помогает обнаружить нелинейность и немонотонность зависимости.

Анализ корреляции позволяет количественно оценить указанное выше соотношение параметров и характеристик производительности. Для такого анализа используется коэффициент корреляции Спирмена, так как он позволяет провести оценку при отсутствии информации о распределении величин.

Нелинейность регрессионной модели описывается соотношением

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,1} x_{i,2} + e_i ,$$

где β_i – соответствующий регрессионный коэффициент, e_i – независимый шум с нулевым ожиданием и постоянной вариацией, а эффекты $x_{i,1}$ и $x_{i,2}$ на результат зависят друг от друга, поэтому не могут быть разделены, и вводится новая величина $x_{i,3} = x_{i,1}x_{i,2}$. Регрессия производится на частично-полиномиальных кривых, представляющих собой ограниченные кубические сплайны с линейными крайними частями. Используется метод наименьших квадратов.

2.3 Нейронные сети и другие методы машинного обучения

Использование нейронных сетей и других методов машинного обучения позволяет выявить неочевидные зависимости, вследствие чего точность предсказаний может повыситься. Но у таких подходов есть и свои недостатки: неявный вид полученных результатов не позволяет делать заключения, основанные на конкретной форме получаемой модели, а для корректного обучения требуется предоставить сравнительно большую тестовую выборку, что особенно сложно в условиях вычислений на реальных суперкомпьютерах.

Метод с использованием нейронных сетей для предсказания времени выполнения параллельных приложений, протестированный на суперкомпьютерах, рассматривается в [11]. Подход, описанный в данной статье, проверяется на примере одного конкретного приложения, хотя и не опирается на его специфические характеристики и теоретически может быть широко применимым.

Из всего пространства параметров равномерно выбираются точки, для которых будут произведены запуски приложения. После обучения модель предоставляет предсказания для точек на всём пространстве параметров. Используются трехслойная полносвязная нейронная сеть прямого распространения с сигмоидой в качестве функции активации, размером скрытого слоя, равным 16, и обратное распространение ошибки. Метод обучения с обратным распространением ошибки оценивает абсолютную ошибку, а не относительную. Для решения этой проблемы используются стратификация и бэггинг. Для тестирования модели отбиралось 1-1.3 тыс. точек из пространства параметров. Использование для тестирования и проверки предсказаний 2-3 тыс. точек позволяет получить предсказания, имеющие относительную ошибку 5-7%, тогда как при использовании 250-500 точек модель имеет ошибку примерно в два раза больше.

Нейронная сеть с такой же конфигурацией используется для построения предсказаний авторами [4], помимо уже описанного выше подхода, основанного на линейной регрессии. При этом регрессия дает несколько более точный результат при предсказании масштабируемости (медианная и средняя относительная ошибка для нейронных сетей соответственно 7.5% и 12.8%). Такая же нейронная сеть применяется и в уже рассмотренной выше статье [10]. Используются обратное распространение ошибки с автоматическим вычислением скорости обучения и момента (resilient backpropagation) и кроссвалидация. Это подход не требует дополнительного анализа данных в сравнении с описанным в этой же статье регрессионным методом. Несколько другая нейронная сеть используется авторами [12] для моделирования времени выполнения приложений на грид-системах. Рассматривается трехслойная искусственная нейронная сеть с радиальной базисной функцией в качестве функции активации. Средняя относительная ошибка предсказаний для разных приложений составила 10-12%.

Авторы [13] осуществляют предсказание энергопотребления и производительности. Предсказание происходит для различных параметров и метрик производительности CUDA-приложений (выборка производится равномерно по всему множеству возможных значений), используются динамические регрессионные модели, строящиеся методами машинного обучения (случайные леса, метод опорных векторов).

2.4 Симуляция исполнения программы

Еще одно направление среди методов предсказания масштабируемости программ – это симуляция их исполнения. Такой подход является наиболее сложным в реализации и требовательным к наличию информации о структуре программы, но примеры реализации моделирования для таких подходов существуют: моделирование может производиться с использованием марковских моделей [14] или частичного детерминированного воспроизведения [15].

3. Предлагаемые методы

Разработка методов прогнозирования масштабируемости производится в соответствии с поставленными нами требованиями универсальности, то есть отсутствия ограничений, явно накладываемых на вычислительную систему или программу, а также требованиями гипотетической закрытости информации о коде самой программы и невозможности его модифицировать, то есть при использовании метода может быть доступен только исполняемый файл.

В таких условиях для построения модели необходимо использовать какие-либо эмпирические данные, позволяющие учитывать в модели те сведения о программе, которые можно получить, не нарушая поставленные требования. Предлагается использовать в качестве таких данных динамические характеристики, полученные для нескольких конкретных запусков приложения (такие запуски будем называть *тестовыми запусками*).

3.1 Произведение тестовых запусков

Обычно суперкомпьютеры предоставляют возможность запуска задач большому количеству пользователей. При этом сразу несколько программ могут исполняться одновременно, если это позволяют ресурсы. Обычно пользователь может осуществить запуск задачи, требующей вычисления на малом количестве узлов, быстрее, чем осуществить запуск задачи на большом количестве узлов. Также большие конфигурации суперкомпьютера могут быть временно недоступны или доступны только ограниченному привилегированному набору пользователей. Исходя из этого актуальную задачу прогнозирования масштабируемости на основе результатов тестовых запусков можно сформулировать как задачу предсказания значений характеристик исполнения приложения на больших конфигурациях вычислительной системы с использованием результатов запусков этого приложения на малых конфигурациях.

В данной статье в качестве динамической характеристики запусков используется время исполнения приложения. Также вследствие особенности результатов, описанных далее в разделе 4, рассматривается не просто время исполнения, а достигаемый минимум времени исполнения приложения на каждом фиксированном наборе параметров на конкретной конфигурации. Далее такой минимум иногда называется просто временем исполнения.

3.2 Тип модели предсказаний

Модели предсказаний можно условно разделить на три категории: модели, позволяющие осуществить экстраполяцию; модели, описывающие явную зависимость от характеристик тестовых запусков; модели с неявными зависимостями.

Моделями, позволяющими осуществить экстраполяцию, будем называть модели, описывающие зависимость значений динамической характеристики $T(q)$ запуска приложения на q процессах от количества процессов q. Здесь для подбора параметров функции $T(q)$ используются результаты k тестовых запусков на $p_1, \ldots, p_k$ процессах. То есть в качестве метода подбора коэффициентов может использоваться регрессия по набору точек $\{(p_i, T(p_i)), i = 1..k\}$. Отнесем также к этому же типу более общие модели, описывающие зависимость $T(q, inp_1, \ldots, inp_m)$ от количества процессов q при запусках с входными параметрами $inp_1, \ldots, inp_m$.

Основная задача заключается в выборе вида функции $T(q)$. Исходя из общих сведений о динамике времени исполнения приложения при увеличении числа процессов можно сделать несколько предположений. Функция должна позволять описывать как случаи хорошо распараллеливаемых приложений (когда время исполнения уменьшается с увеличением количества процессов), так и случаи плохой распараллеливаемости (когда время исполнения в какой-то момент начинает расти по мере увеличения количества процессов, даже если сначала время падало). При этом даже для хорошо распараллеливаемого приложения при гипотетическом бесконечном росте размера конфигурации вычислительной системы в какой-то момент количество вычислений, приходящееся на один процесс, будет занимать малое время по сравнению с накладными расходами, необходимыми для инициализации, синхронизации и коммуникаций на большом числе процессов. Из-за накладных расходов прирост получаемого ускорения обычно сначала уменьшается, а потом становится отрицательным.

Это означает, в частности, что производная функции экстраполяции T (q) отрицательна, пока время исполнения уменьшается, и становится в какой-то момент положительной, если время исполнения в какой-то момент начинает расти. Такими свойствами может обладать, например, производная функции $T(q) = aq + bq^{-1} + cq^{-1/2}$: $T^{(1)}(q) = a - (2b + cq^{1/2}) / 2q^2$. Подбор регрессионных коэффициентов функции T (q) может осуществляться при помощи регрессии с использованием метода наименьших квадратов по алгоритму Левенберга-Марквардта [16, 17]. В разделе 4 представлены результаты тестирования этой модели.

Модели данного типа описывают простую зависимость и предоставляет малую степень гибкости предсказаний. Однако исследования такого типа моделей могут помочь подобрать вид функции, которую впоследствии можно использовать в моделях другого типа.

Моделями, описывающими явную зависимость от характеристик тестовых запусков, будем называть модели, описывающие зависимость значений динамической характеристики $T(t_1, ..., t_k, q)$ запуска приложения на q процессах от q и $\{t_i\}$ – значений динамической характеристики исполнения программы на $\{p_i\}$ процессах, i = 1..k. Как и в предыдущем случае, более общую модель с функцией $T(t_1, ..., t_k, q, inp_1, \ldots, inp_m)$ также будем относить к моделям, описывающим явную зависимость. Простым примером такой модели может быть модель с линейной зависимостью по переменным: $T(t_1, ..., t_k, q) = a_1t_1 + \ldots + a_kt_k + bq$. Подбор коэффициентов может быть осуществлен с помощью множественной линейной регрессии по аналогии с [3]. Здесь рассматривается применимость использованного авторами статьи подхода, поэтому сама задача ставится по-другому – как задача интерполяции.

Наконец, к моделям с неявными зависимостями можно отнести построение предсказателей, описывающих сложно формализуемую зависимость, с использованием нейронных сетей и прочих методов машинного обучения. На данном этапе такие модели не тестировались из-за недостаточно большого объема имеющихся данных по запускам, но в будущем применение этих моделей для решения поставленной задачи рассматривается.

3.3 Ошибка предсказаний

Точность предсказаний моделей оценивается с использованием относительной ошибки. Время работы эффективно написанных параллельных программ может значительно уменьшиться с ростом числа процессов, на которых она запущена. Поэтому данные, используемые для построения и проверки прогноза, основанного на времени работы приложения, в некоторых случаях могут иметь значительный разброс. В то же время многие методы, используемые для подбора регрессионных коэффициентов, минимизируют абсолютную ошибку, а не относительную. Абсолютная ошибка аппроксимации времени вычисления на малом количестве процессов может быть большой при малой относительной ошибке, а на большом количестве процессов – малой при большой относительной ошибке. Поставленная в данной работе задача подразумевает большую желаемую точность предсказаний именно для большего числа процессов. Отсюда возникает проблема ориентирования аппроксимации на малые конфигурации.

На самом деле, такая вынужденная перестановка приоритетов не обязательно имеет место: практические результаты показывают, что, например, форма выбранных для аппроксимации функций этому препятствует. Данную проблему можно решить переходом от предсказания времени исполнения к предсказанию другой характеристики, имеющей меньший разброс значений, исключением результатов некоторых запусков на самых малых конфигурациях, а также использованием регрессионных методов с весами.

3.4 Общая схема работы предлагаемых методов

Для выбранной модели предсказаний работу метода можно разделить на несколько этапов. На первом этапе производятся множественные тестовые запуски конкретного приложения на малых конфигурациях, включая идентичные запуски, и происходит сбор данных о динамических характеристиках таких запусков (в данной работе единственная динамическая характеристика – это время исполнения). На втором этапе собранные данные обрабатываются: выбирается минимум времени исполнения по идентичным запускам, данные приводятся к единому виду. На третьем этапе данные, полученные на предыдущем этапе, используются для подбора пара-

метров выбранной модели предсказаний, в результате чего получается готовая модель с фиксированными значениями параметров. На четвертом этапе происходит построение предсказаний времени исполнения приложения для заданной целевой конфигурации с помощью готовой модели.

Стоит отметить, что общий вид модели предсказаний в этом случае одинаковый для всех приложений, но параметры этой модели подбираются отдельно для каждого приложения по конкретному набору его тестовых запусков.

Для проверки корректности предсказаний совершается несколько идентичных запусков для каждой целевой конфигурации, для каждого набора идентичных запусков выбирается минимум времени исполнения. Эти значения минимумов времени исполнения на целевых конфигурациях используются для вычисления относительной ошибки предсказаний модели.

4. Проверка применимости методов

Проверка применимости разрабатываемых методов осуществлялась на суперкомпьютерах «Ломоносов» и «Ломоносов-2» с использованием реализации HPL теста Linpack и теста BT из NPB 3.3.1. Было проведено исследование сильной масштабируемости этих приложений при нескольких фиксированных размерах матриц.

Вычислительные узлы суперкомпьютера «Ломоносов» включают два процессора Intel Xeon X5570 (2.93ГГц) и связаны коммуникационной сетью QDR InfiniBand 4x. Компиляция производилась при помощи mpicc. Вычислительные узлы суперкомпьютера «Ломоносов-2» включают процессор Intel Haswell-EP E5-2697v3 (2.6 ГГц) и связаны коммуникационной сетью Infiniband FDR. Компиляция производилась при помощи mpicc и mpif77.

При исследовании масштабируемости HPL для сбора данных производились множественные тестовые запуски на конфигурациях с числом процессов, равным 4n, n = 1..32, и несколько запусков на конфигурациях со 144, 256, 400, 576, 784, 1024 процессами (размеры конфигураций выбраны квадратами целых чисел для исключения несбалансированности, возможной из-за изменения соотношения сторон процессорной сетки) при фиксированных размерах матрицы, равных 16384, 25600, 50000.

Для исследования BT тестовые запуски производились на конфигурациях с числом процессов, равным n^2, n = 2..11, а также производились запуски на конфигурациях со 144, 256, 400, 576, 784, 1024 процессами. Все запуски BT проходили при фиксированных размерах матрицы, равных 64^3 и 162^3, что соответствует задачам классов A и C для NPB. Стоит отметить, что для задачи класса A запуски на 576, 784, 1024 процессах являются некорректными, поэтому построение предсказаний в этом случае становится невозможным.

Результаты повторных запусков на одной и той же конфигурации показали наличие сильных разбросов значений времени исполнения. Эта особенность проиллюстрирована на Рис. 1, где отображены все проведенные запуски, включая идентичные, для теста Linpack на суперкомпьютере «Ломоносов» при размере матрицы, равном 25600.

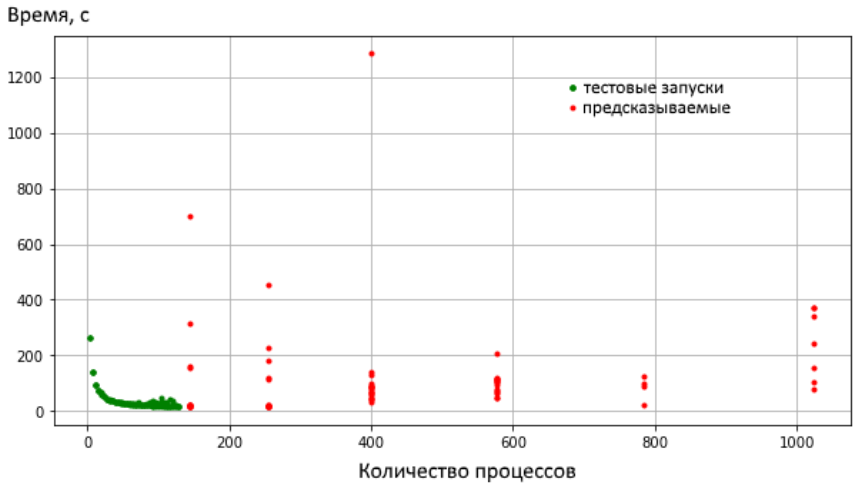

Рис. 1. Время исполнения Linpack на суперкомпьютере «Ломоносов», размер матрицы равен 25600

Максимальный разброс времени исполнения для идентичных запусков в отдельных случаях доходит до 4188%. Такое изменение времени исполнения можно объяснить наличием изменений в нагрузке коммуникационной сети, а также возможным дисбалансом в работе различных вычислительных узлов, потенциальными отказами аппаратуры и различным размещением процессов по вычислительным узлам.

С учетом полученной дисперсии времени исполнения было принято решение выбирать минимум времени исполнения по идентичным запускам. Предсказания строились также для соответствующих минимумов времени исполнения. На конфигурациях, используемых для построения предсказаний (до 128 процессов), учитывались результаты 3-4 идентичных тестовых запусков. На конфигурациях, для которых строились предсказания (144-1024 процесса), учитывались результаты 8-25 идентичных запусков.

Для более точного учета результатов тестовых запусков в разделе 4.1 размер конфигурации был ограничен снизу 16 процессами.

4.1 Результат тестирования экстраполяционной модели

Рассматривалась предложенная в разделе 3.2 функция $T(q) = aq + bq^{-1} + cq^{-1/2}$ при a, b, c >= 0. В Табл. 1 и Табл. 2 представлены значения соответствующих относительных ошибок для предсказанного времени исполнения Linpack и BT соответственно.

Таблица 1. Относительные ошибки предсказания времени исполнения Linpack

	144	**256**	**400**	**576**	**784**	**1024**
Lomonosov, matr_size = 16384	5.14%	6.20%	55.30%	21.92%		
Lomonosov, matr_size = 25600	1.10%	38.94%	7.65%	22.04%		15.24%
Lomonosov-2, matr_size = 25600	10.83%	10.40%	11.46%	18.05%	10.82%	9.67%
Lomonosov-2, matr_size = 50000	9.95%	4.17%	0.75%	5.76%	7.72%	

Таблица 2. Относительные ошибки предсказания времени исполнения BT

	144	**256**	**400**	**576**	**784**	**1024**
Lomonosov-2, class A	10.47%	18.64%	31.50%	x	x	x
Lomonosov-2, class C	1.39%	6.01%	22.69%	5.70%	19.20%	22.31%

Пример графика, аппроксимирующего время исполнения Linpack (запуски проводились на суперкомпьютере «Ломоносов-2» при фиксированном размере матрицы 25600), приведен на Рис. 2.

Стоит отметить, что аппроксимация функцией T(q) на самих конфигурациях с числом процессов 144, 256, 400, 576, 784, 1024 для Linpack все равно дает относительную ошибку, которая может доходить до 15%, что накладывает ограничения на допустимую точность предсказаний с

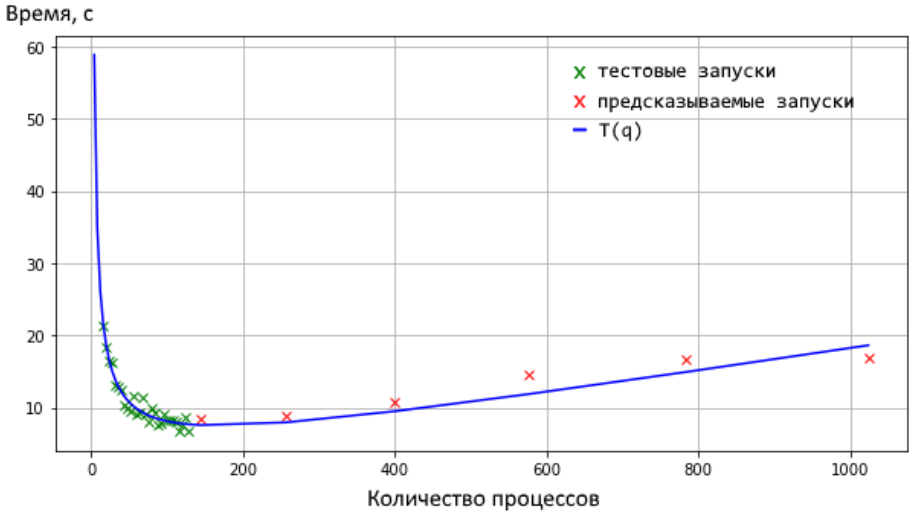

Рис. 2. График экстраполирующей функции T (q) для времени исполнения Linpack

использованием функции такого вида. В этом смысле предсказания времени исполнения BT такой функцией T(q) возможны с меньшими ошибками, но из-за более сильного ограничения на количество тестовых запусков (размеры конфигураций обязаны быть квадратами целых чисел) влияние несбалансированности ошибки предсказаний, о котором говорилось в разделе 3.3, становится более заметным.

4.2 Результат тестирования модели с явной зависимостью

Была предпринята попытка построения прогноза для Linpack с помощью линейной модели, описывающей явную зависимость от результатов тестовых запусков и имеющей вид T (t_1, ..., t_k, q, matrix_size), t_n – время выполнения на p_i процессах, i = 4n, n = 1..k, k = 32, q ∈ {144, 256, 400, 576, 784, 1024}, matrix_size ∈ {16384, 25600, 50000}. Предсказания, полученные с помощью данного подхода, имели большую относительную ошибку (>60%) и оказались некорректными. Это может быть объяснено в частности недостаточно большим набором тестовых данных при большом количестве регрессионных коэффициентов. Для уменьшения числа параметров можно использовать анализ корреляции и выбирать наиболее коррелирующие с временем исполнения параметры.

В данной модели вид зависимости от количества процессов q можно представить в виде слагаемого, имеющего форму подходящей функции, описанной в разделе 4.1. Отчасти такой подход отражает описанный в [3] вид моделирующей функции.

4.3 Сравнение точности предсказаний

Точность предсказаний времени исполнения для теста Linpack на суперкомпьютерах «Ломоносов» и «Ломоносов-2», описанных в разделе 4.1, сравнивается с предсказаниями для теста Linpack, описанными в статье [10]. При этом авторы [10] ставят задачу интерполяции, а в разделе 4.1 рассматривается задача экстраполяции.

Таблица 3. Медианные относительные ошибки предсказаний времени исполнения Linpack

Частично-полиномиальная регрессия [10]	**Нейронные сети [10]**	**Предложенный подход**
2.2 – 5.2% (до 512 процессов)	6.5 – 10.5% (до 512 процессов)	5.76 – 15.24% (до 1024 процессов) 4.17 – 10.83% (до 512 процессов)

В Табл. 3 приводятся соответствующие диапазоны значений медианных относительных ошибок. Ошибка предсказаний, реализованных в данной работе, сравнима с представленными в [10] значениями для запусков на количестве процессов до 512. Улучшить предсказания можно, например, коррекцией метода, позволившей бы точнее учитывать влияние больших абсолютных ошибок при малых относительных, как описано в разделе 3.3, или использованием модели с большим числом параметров, такой как в разделе 3.2.

5. Заключение

В данной работе представлены описания разрабатываемых методов прогнозирования масштабируемости параллельных приложений, основанные на эмпирических данных, и произведен обзор существующих подходов к решению задач предсказания масштабируемости. Предложенные нами методы основываются на использовании результатов тестовых запусков приложения на малых конфигурациях вычислительной системы для предсказания значений характеристик исполнения этого приложения на конфигурациях большего размера.

Применимость методов была оценена с помощью запусков на суперкомпьютерах «Ломоносов» и «Ломоносов-2» на примере реализации теста Linpack и теста NAS BT. Исследовалась сильная масштабируемость, а в качестве характеристики масштабируемости было ис-

пользовано время выполнения приложения. Медианные относительные ошибки предсказаний, построенных с помощью экстраполирующей модели, составили 5.76 – 18.64%.

Дальнейшее направление исследований включает в себя продолжение разработки методов с моделями, описывающими явную зависимость от результатов тестовых запусков, и разработку методов с моделями, описывающими неявную зависимость.

Литература

1. Воеводин Вл.В., Жуматий С.А., Соболев С.И., Антонов А.С., Брызгалов П.А., Никитенко Д.А., Стефанов К.С., Воеводин Вад.В. Практика суперкомпьютера "Ломоносов" // Открытые системы. СУБД. 2012. № 7. С. 36–39.
2. Antonov A., Teplov A. Generalized Approach to Scalability Analysis of Parallel Applications // Lecture Notes in Computer Science. 2016. Vol. 10049. P. 291–304.
3. Barnes B. J., Rountree B., Lowenthal D. K., Reeves J., de Supinski B., Schulz M. A regression-based approach to scalability prediction // Proceedings of the ICS. 2008. P. 368-377.
4. Singh K., Curtis-Maury M., McKee S. A., Blagojević F., Nikolopoulos D. S., de Supinski B. R., Schulz M. Comparing Scalability Prediction Strategies on an SMP of CMPs // Euro-Par Parallel Processing. Lecture Notes in Computer Science. 2010. Vol. 6271. P. 143-155.
5. Marathe A., Harris R., Lowenthal D. K., de Supinski B. R., Rountree B., Schulz M. Exploiting Redundancy and Application Scalability for Cost-Effective, Time-Constrained Execution of HPC Applications on Amazon EC2 // IEEE Transactions on Parallel and Distributed Systems. 2016. Vol. 27, No. 9. P. 2574-2588.
6. Escobar R., Boppana R. V. Performance Prediction of Parallel Scientific Applications // HPDC Poster. 2017.
7. Chen T.-Y., Khalili O., Campbell R. L., Carrington L., Tikir M. M., Snavely A. Performance Prediction and Ranking of Supercomputers // Advances in Computers. 2008. Vol. 72. P. 135-172.
8. Chatzopoulos G., Dragojević A., Guerraoui R. ESTIMA: Extrapolating ScalabiliTy of In-Memory Applications // ACM Transactions on Parallel Computing. 2017. Vol. 4, No. 2, Article 10.
9. Escobar R., Boppana R. V. Performance Prediction of Parallel Applications Based on Small-Scale Executions // Proceedings of the HiPC. 2016. P. 362-371.
10. Lee B. C., Brooks D. M., de Supinski B. R., Schulz M., Singh K., McKee S. A. Methods of Inference and Learning for Performance Modeling of Parallel Applications // Proceedings of the PPoPP. 2007. P. 249-258.
11. Ipek E., de Supinski B., Schulz M., McKee S. A. An approach to performance prediction for parallel applications // Euro-Par Parallel Processing. Lecture Notes in Computer Science. 2005. Vol. 3648. P. 196-205.
12. Nadeem F., Alghazzawi D., Mashat A., Fakeeh K., Almalaise A., Hagras H. Modeling and predicting execution time of scientific workflows in the Grid using radial basis function neural network // Cluster Computing. 2017. Vol. 20, No. 3. P. 2805-2819.
13. Rejitha R. S., Shajulin Benedict, Suja A. Alex, Infanto S. Energy prediction of CUDA application instances using dynamic regression models // Computing. 2017. Vol. 99, No. 8. P. 765-790.
14. Grobelny E., Bueno D., Troxel I., George A. D., Vetter J. S. FASE: A Framework for Scalable Performance Prediction of HPC Systems and Applications // SIMULATION. 2007. Vol. 83, No. 10. P. 721-745.
15. Zhai J., Chen W., Zheng W., Li K. Performance Prediction for Large-Scale Parallel Applications Using Representative Replay // IEEE Transactions on Computers. 2016. Vol. 65, No. 7. P. 2184-2198.

16. Levenberg K. A Method for the Solution of Certain Non-Linear Problems in Least Squares // Quarterly of Applied Mathematics. 1944. Vol. 2, No. 2. P. 164-168.

17. Marquardt D. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters // SIAM Journal on Applied Mathematics. Vol. 11, No. 2. P. 431-441.