# Параллельная обработка данных сейсморазведки с использованием расширенной модели Master-Slave

А.П. Бурцев[1]

МГУ имени Ломоносова[1]

Поиск нефтяных и газовых месторождений – важная задача для современного общества. Высокая стоимость бурения и проведения поисковых работ делает задачи обработки и анализа данных сейсморазведки особенно важными и актуальными. Учитывая большой объём собираемых данных и высокую вычислительную сложность алгоритмов обработки, для эффективного решения поставленных задач необходимо использование высокопроизводительных систем. В статье предлагается расширение модели "Master-Slave" для организации параллельной обработки данных, с учётом особенностей задач сейсморазведки. Предлагаемый подход облегчает перенос последовательных алгоритмов на параллельные вычислительные системы, обеспечивая высокую масштабируемость, возможность динамической балансировки нагрузки и учёт специфики гетерогенных вычислительных кластеров.

*Ключевые слова:* Параллельные вычисления, сейсморазведка, гетерогенные кластеры, масштабируемые параллельные алгоритмы.

## 1. Введение

Поиск и добыча полезных ископаемых играет важную роль в жизни и развитии современного общества. Особенно сильно мировые экономики зависят от запасов углеводородов – нефти и газа. Так, по данным Мирового энергетического агентства (International Energy Agency) за 2012 год производство нефтепродуктов в мире составляло 91.3 миллионов баррелей в сутки, а потребление 90.0 миллионов баррелей. Высокие риски и стоимость геологоразведочных работ вынуждают специалистов, занимающихся обработкой результатов полевых наблюдений, постоянно совершенствовать и расширять существующие базы алгоритмов обработки данных. Учитывая большой объём геофизических данных, использование высокопроизводительных систем и параллельных алгоритмов – неотъемлемая часть их анализа. В качестве примера рассмотрим месторождение Карачаганак, 3D съёмка которого была проведена в 2009 году: площадь наблюдений составила примерно 900 тысяч квадратных километров, а объём собранных данных более 100 терабайт.

Для обработки данных сейсморазведки используют различные математические методы: цифровую обработку сигналов, численное моделирование, математическую статистику и теорию случайных процессов. Чаще всего разработкой таких алгоритмов занимаются специалисты в области физики земли, геологии и сейсмики. Таким образом, процесс разработки программных решений часто ложится на плечи людей, для которых программирование не является основной специальностью. В связи с этим, актуальным является разработка методов организации параллельных вычислений, упрощающих построение параллельных программ для обработки сейсмических данных. В данной статье будут рассмотрены современные подходы к организации параллельных вычислений и представлено расширение классической модели Master-Slave, адаптированное для решения поставленных задач. Предложенный подход учитывает гетерогенную структуру вычислительных систем и при этом достаточно прост для программиста. Конечно, подобный подход применим лишь к задачам, имеющим определённую структуру. Поэтому в работе также будет уделено внимание описанию наиболее типичных задач, характерных для сейсморазведки, и методов их решения.

# 2. Особенности задач сейсморазведки

Задачи обработки данных сейсморазведки используют множество различных алгоритмов, но в них можно попытаться выделить общие особенности организации вычислений и методов хранения данных. Опираясь на них, можно разработать обобщённый масштабируемый параллельный метод обработки данных, который позволит решать поставленные задачи. Кроме того, необходимо рассмотреть существующие системы обработки сейсмических данных. В качестве такого примеры была выбрана система SeisSpace ProMax, разработанной компанией Halliburton.

## 2.1 Сейсмические данные и алгоритмы их обработки

Сейсморазведка – раздел прикладной геофизики, основанный на анализе результатов регистрации искусственно возбуждаемых упругих волн, и извлечении из этих данных полезной геолого-геофизической информации. За регистрацию данных отвечают сейсмоприёмники – приборы для преобразования механических колебаний в электрический сигнал – ток переменного напряжения. Последовательность значений амплитуд колебаний тока, полученная на сейсмоприемнике, называется сейсмической трассой. Сейсмические трассы объединяются в сейсмограммы по некоторому признаку, например, общему пункту приёма (ОПП), общему пункту возбуждения (ОПВ) или общей глубиной точке (ОГТ) [1]. Таким образом, кроме самой сейсмической трассы обычно необходимо хранить и дополнительную информацию: номера и координаты пунктов возбуждения и пунктов приёма, даты проведения работ, частоту дискретизации с которой работали приёмники и многое другое. Такую техническую информацию принято называть заголовком сейсмической трассы.

С точки зрения вычислительной системы можно выделить два принципиально разных способа хранения информации. Тела трасс и заголовки можно разместить в одной или в различных областях памяти. Первый способ обеспечивает более удобный способ обработки данных, так как заголовки меняются значительно реже, чем тело трассы, а большинство алгоритмов предполагают, что данные представлены в виде многомерной матрицы. Однако, возникает необходимость ассоциировать тело и заголовок трассы, что может быть не всегда удобно. Второй подход не требует специального сопоставления тел трасс и их заголовков, но усложняет доступ к данным в удобном для обработки виде.

В формализованном виде большинство алгоритмов обработки данных сейсморазведки можно записать следующим образом: $R(\bar{x}', t) = f(D(\bar{x}, t))$, где $\bar{x}$ – вектор входных координат, $\bar{x}'$ – вектор выходных координат, которые определяют положение трасс, $t$ – время, $D(\bar{x}, t)$ – входные данные, $R(\bar{x}, t)$ – результат обработки, $f(D)$ – функция обработки. При этом сейсмические данные (как входные, так и результат обработки) обычно можно представить в матричном виде. Так, например, двумерная задача многоканальной фильтрации может быть записана в следующем виде: $y = w * z$, где $w$ – двумерная матрица-фильтр, $z$ – двумерная матрица входных трасс, $y$ – двумерная матрица – результат обработки, $*$ – оператор матричной свёртки. В качестве другого примера рассмотрим задачу многомерной интерполяции. Её цель – построение трасс на новой координатной сетке. Для этого используется информация о соседних трассах и различные геофизические предположения. Несмотря на то, что такой алгоритм не всегда можно представить в матричном виде, данные всё равно удобно хранить в виде многомерной матрицы.

Важнейшая особенность обработки связана с тем, что в большинстве алгоритмов присутствует независимость по данным. Именно это будет полезно при построении обобщённого параллельного алгоритма. Обычно независимыми являются разные сейсмограммы. Таким образом, имея последовательный алгоритм и систему распределения данных, можно решать поставленную задачу на высокопроизводительной вычислительной системе, используя естественный параллелизм по данным.

### 2.2 Система SeisSpace ProMax

Подобный подход обработки данных применяется в коммерческой системе SeisSpace ProMax, разработанной компанией Halliburton. Она является кроссплатформенной за счёт использования языка Java. В неё включены: система хранения, обработки и визуализации данных.

Система хранения состоит из: базы данных (database), наборов данных (dataset), потоков обработки (dataflow), а также других вспомогательных файлов. База данных содержит в себе общую информацию для текущего проекта, например, описание геометрии наблюдений. Наборы данных содержат сейсмические трассы и их заголовки, объединённые в гиперкуб. В рамках одного проекта может содержатся множество наборов данных, привязанных к одной базе данных. Поток обработки содержит в себе последовательность модулей (отдельных программ обработки) и их параметры, таким образом, процесс обработки данных становится более простым и наглядным.

Система обработки отвечает за параллельную обработку данных в рамках одного модуля и конвейерную обработку между модулями в рамках одного потока. Также существует возможность построения сложных параллельных программ с использованием некоторых особенностей технологии MPI в рамках одного модуля. Но отметим, что использование MPI технологии требует от разработчика полноценного написания кода.

Система визуализации обеспечивает удобный доступ к данным и позволяет проводить некоторые операции по анализу результатов обработки в интерактивном режиме.

SeisSpace ProMax позволяет пользователю легко расширять список используемых модулей. Для этого необходимо разработать два класса на языке Java. Первый отвечает за интерактивное меню и список параметров для встраивания в интерактивную систему обработки. Второй отвечает за решение самой задачи и опирается на параметры, которые передаются через меню и данные, включённые в систему хранения. Важнейшим является тот факт, что пользователю не нужно, в большинстве случаев, заниматься процессом построения параллельного алгоритма. Этим, при условии наличия параллелизма по данным, будет заниматься сама система. Таким образом, встраивать новые модули достаточно просто, а использование кроссплатформенного языка обеспечивает высокую переносимость. Отметим, что частичная поддержка технологии MPI позволяет создавать и встраивать модули практически любой сложности.

Несмотря на все свои достоинства, система SeisSpace ProMax не лишена некоторых недостатков. Выделим ключевые:

1. С точки зрения системы, все узлы вычислителя являются равнозначными, таким образом, если ресурсов недостаточно на одном из них, то задача, возможно, не будет выполнена целиком.
2. Если среди данных присутствуют некорректные (которые вызовут ошибку при обработке), то такая задача не будет иметь решения.
3. Без использования технологии MPI невозможно проводить как предварительную, так и заключительную обработку данных, используя при этом автоматические методы распараллеливания.

## 3. Предлагаемый метод обработки данных

Была поставлена задача разработать метод автоматического распределения работы, который поможет упростить процесс создания параллельных программ на основе существующих последовательных алгоритмов. Для этого рассмотрим существующие подходы, опирающиеся на параллелизм по данным. Среди которых: популярная технология MapReduce для обработки данных большого объёма, технология Microsoft Dryad основанная на представлении программы в виде ациклического графа, представляющая подход Data Flow, и классический вариант модели Master-Slave. Предложенная модель распределённой обработки данных учитывает достоинства и недостатки, рассмотренных подходов.

### 3.1 MapReduce, Dryad и Master-Slave

Подробно проекты MapReduce и Dryad были рассмотрены в статье [2].

Функция высшего порядка — в программировании функция, принимающая в качестве аргументов другие функции или возвращающая другую функцию в качестве результата. Основная идея состоит в том, что функции имеют тот же статус, что и другие объекты данных. Использование функций высшего порядка приводит к абстрактным и компактным программам, принимая во внимание сложность производимых ими вычислений. Map — функция высшего порядка, которая применяет заданную функцию к каждому элементу списка, возвращая список результатов. Свёртка списка (англ. folding, также известна как reduce или accumulate) — функция высшего порядка, которая производит преобразование структуры данных к единственному атомарному значению при помощи заданной функции. Операция свёртки часто используется в функциональном программировании при обработке списков [3].

MapReduce [4] – модель программирования и платформа для пакетной обработки больших объемов данных, разработанная и используемая внутри компании Google. Технология MapReduce основана на сочетании двух основных операций Map и Reduce. В рамках этой парадигмы все вычислительные узлы разбиваются на главный (или главные) и рабочие узлы. На Map-шаге происходит предварительная обработка входных данных. Для этого главный узел получает входные данные, разделяет их на части и передает рабочим узлам для предварительной обработки. Рабочие узлы на этом шаге, собственно, проводят предварительную обработку данных. Название данный шаг получил от одноименной функции высшего порядка. На Reduce-шаге происходит свёртка предварительно обработанных данных. Главный узел получает ответы от рабочих узлов и на их основе формирует результат — решение задачи, которая изначально формулировалась.

Преимущество MapReduce заключается в том, что он позволяет распределено производить операции предварительной обработки и свертки. Операции предварительной обработки работают независимо друг от друга и могут производиться параллельно (хотя на практике это ограничено источником входных данных и/или количеством используемых процессоров). Аналогично, множество рабочих узлов могут осуществлять свертку: для этого необходимо только, чтобы все результаты предварительной обработки с одним конкретным значением ключа обрабатывались одним рабочим узлом в один момент времени. Хотя этот процесс может быть менее эффективным по сравнению с другими последовательными алгоритмами, MapReduce может быть применен к большим объёмам данных, которые могут обрабатываться большим количеством вычислительных узлов. Используемая в Google реализация MapReduce является закрытой технологией, однако существует общедоступная реализация Apache Hadoop. К её достоинствам следует отнести повышенную надёжность, система способна проводить автоматический перезапуск заданий на рабочих узлах в случае их отказов. При использовании нескольких главных узлов, так же возможна и их замена во время выполнения.

Универсальная технология распределенной обработки данных Dryad разрабатывается компанией Microsoft и используется в поисковой системе MSN Live Search. В настоящее время она доступна для некоммерческого использования. Описание технологии было опубликовано в работе [5]. Модель программирования Dryad основана на представлении приложения в виде ориентированного ациклического графа. Вершинами графа являются процессы, а рёбра графа определяют потоки данных между процессами в виде односторонних каналов. У процесса может быть несколько входных и выходных каналов.

Важно отметить, что модель программирования Dryad содержит в себе, в качестве частных случаев, реляционную алгебру и MapReduce. Система организует выполнение приложения на имеющихся вычислительных ресурсах, будь то многоядерная машина или кластер из большого числа узлов. При этом система автоматически осуществляет планирование вычислений, распределение вершин между машинами, обработку отказов и динамическую оптимизацию структуры графа. Таким образом, основной сложностью при построении параллельных программ является построение ацикличного графа их выполнения.

В рамках модели Master-Slave [6] все узлы разбиваются на две группы: мастера (master) и работники (slave). Мастер отвечает за распределение работы, он посылает задания (независимые части решаемой задачи), а работники выполняют обработку и возвращают результат мастеру. Такой подход обеспечивает простой метод организации балансировки нагрузки в гетерогенной системе и позволяет контролировать статус завершения задания, и таким образом, обеспечивать надёжность вычислений.

### 3.2 Основные положения расширенной модели Master-Slave

Учитывая особенности задач сейсморазведки, разработанный метод должен обеспечивать:

1. Классификацию узлов вычислительной системы по доступным ресурсам;
2. Распределение работ в автоматическом режиме;
3. Перезапуск не выполненных задач на узлах другого класса;
4. Сбор статистики по результатам работы;
5. Предварительную обработку и обобщение результатов на мастер узле;
6. Поддержка перезапуска невыполненных задач с изменёнными параметрами и продолжение работы в случае сбоя в работе вычислительного комплекса.

Поэтому, для решения поставленной задачи было решено разработать расширенную модель Master-Slave. В отличие от методов, основанных на построении графа программ, предложенный подход не требует внесения существенных изменений в структуру программы при добавлении новых алгоритмов. А в отличие от методов, основанных на технологии MapReduce, метод естественным образом позволяет организовать классификацию вычислительных узлов по доступным ресурсам.

Для большинства алгоритмов обработки сейсмических данных наиболее важными ресурсами являются: оперативная память и число вычислительных ядер. Чаще всего именно недостаток оперативной памяти не позволяет решить поставленную задачу на выбранной вычислительной системе. Различное число вычислительных ядер на разных узлах системы чаще всего приводит только к изменению времени выполнения, которое может быть компенсировано системой распределения работ за счёт балансировки нагрузки. Необходимо также отметить, что межу числом используемых вычислительных ядер и требуемым объёмом оперативной памяти существует прямая связь. Обычно каждое ядро использует некоторый набор «личных» данных, которые используются в монопольном режиме, следовательно, чем их больше, тем больше памяти будет занято. Наличие сопроцессора – новая тенденция в построении вычислительной системы. Главная проблема в его использовании – необходимость построения специального программного кода. По этой причине, в данной работе поддержка сопроцессоров не рассматривается. Таким образом при анализе доступных узлов на вычислителе необходимо в первую очередь получить информацию о вычислительных ядрах и объёме доступной оперативной памяти. В качестве наиболее простой классификации, узлы можно разделить на группы по абсолютным значениям этих параметров.

Система автоматического распределения работ основана на принципе обмена сообщениями. Работники посылают мастеру запросы на выдачу работы и результат обработки предыдущего задания. Мастер отвечает на запросы, высылая новые задания. Если задание завершилось ошибкой, мастер пытается передать задание узлу другого класса, на котором запусков ещё не проводилось. Повторные запуски необходимы для решения проблем, связанных с особенностями классов узлов, например, недостаток оперативной памяти. Если все задания завершились успехом или исчерпали запас доступных повторных запусков, то мастер рассылает всем работникам сообщения о завершении работы.

В результате сбора статистики становится возможно поддерживать аналог точек восстановления и совершать перезапуск только незавершённых задач с изменёнными параметрами. Для этого достаточно сохранять статусы выполнения задач на устройстве постоянной памяти, сразу после их получения на мастер узле. Таким образом в любой момент времени на устройстве постоянной памяти будет содержаться список всех завершённых задач и их статус.

Процесс предварительной и пост обработки на мастер узле организуется естественным образом, непосредственно перед отправкой задания или сразу после получения результата. Однако, чем сложнее эти процессы, тем хуже будет эффективность работы системы в целом.

Особенности предложенного метода накладывают ограничения на структуру решаемых задач и вычислительную систему:

1. В задаче должен присутствовать явный параллелизм по данным, например, по сейсмограммам;
2. Алгоритм обработки должен состоять из следующих чётко обособленных частей: фаза инициализации, фаза обработки и фаза обобщения результатов. При этом для эффективной ра-

боты необходимо чтобы инициализация и обобщение занимали значительно меньше времени, чем обработка;
3. Вычислительная система должна поддерживать сбор параметров, необходимых для классификации узлов, а их значения не должны меняться во время выполнения программы.

### 3.3 Практическая реализация расширенной модели Master-Slave

Основываясь на предложенном методе была разработана программная реализация на языке C++. Она представляет собой систему классов в которую входят: основной класс для организации распределённых вычислений и классы содержащие параметры задач и результаты обработки. Основной класс содержит все методы связанные с распределением нагрузки, хранением таблиц задач и результатов, сохранением промежуточных результатов выполнения программы. Фактически именно он отвечает за саму модель параллельных вычислений. Два остальных класса необходимы для организации удобного доступа к параметрам задачи, а также результатам обработки.

Система обработки использует технологию MPI для организации распределения работ за счёт пересылки сообщений. Таким образом для выполнения работы, все MPI-процессы разделяются на мастера и работников. А решение любой практической задачи проводиться в несколько основных этапов:
1. Инициализация MPI;
2. Передача основных параметров, например, из командной строки;
3. Создание таблиц заданий и результатов;
4. Подготовка структуры MPI сообщений;
5. Определение класса процессов-работников;
6. Основной цикл распределения работ;
7. Завершение MPI.

Первый и последний этапы необходимы для использования технологии MPI. Второй позволяет облегчить процесс доступа к параметрам задачи и данным. Подготовка таблицы задач и структуры MPI сообщений необходима для организации процесса распределения работ. Очевидно, что создание структуры сообщений самый сложный процесс с точки зрения программирования. Но если таблицу задач формировать на всех процессах в системе, а результатом выполнения является только статус (номер ошибки, если таковая была), то можно использовать стандартную структуру сообщений. Другим положительным эффектом при таком подходе является снижение нагрузки на коммуникационную сеть. Именно такой вариант построения таблицы задач применяется в разработанной практической реализации. Процесс определения классов процессов позволяет осуществлять перезапуск невыполненных задач и может осуществляться за счёт стандартной процедуры. А основной цикл распределения работ не требует внесения изменений в свою структуру, вне зависимости от решаемой задачи.

Рассмотрим некоторые теоретические оценки. Введём следующие обозначения: $N_S$ – число процессов-работников, $T_w$ – время обработки одного задания, $T_s$ – время необходимое на отправку задания, $T_r$ – время необходимое на получение результатов, $T_m$ – время вынужденного простоя для одного процесса, $N_T$ – число заданий.

В начале обработки лучшее время простоя для процесса можно оценить следующим числом: $N_S \cdot T_S$. После того как все узлы получили задания, время простоя может быть сведено к нулю, при условии, что с некоторого момента $T_w = (N_s - 1) \cdot (T_s + T_r)$. На завершающем этапе минимальное время можно оценить значением: $(N_S - 1) \cdot T_r$. Таким образом в лучшем случае время простоя на одном из узлов будет составлять: $T_m = N_S \cdot T_S + (N_S - 1) \cdot T_r$. Учитывая тот факт, что время обработки всех задач можно оценить снизу значением: $(T_w \cdot N_T) / N_S$, минимальное время работы программы составит: $N_S \cdot T_S + (N_S - 1) \cdot T_r + (T_w \cdot N_T) / N_S$.

Для повышения надёжности экспериментов, вместо реальных, были выбраны синтетические задачи: каждый процесс-работник вместо расчётов простаивает некоторое (фиксирован-

ное) время. Таким образом можно изучить поведение программы в различных режимах, опираясь на известные характеристики запуска: число задач, время выполнения одной задачи, число процессов-работников. Для максимальной наглядности время выполнения одной задачи было выбрано в диапазоне от 1 мс. до 100 мс. Именно на таких, сверхкоротких, задачах легче всего продемонстрировать основные достоинства и недостатки предложенного подхода.

В таблицах приводятся результаты выполнения программы. $T_{общ}$ – общее время выполнения программы. $T_{мп}$ – максимальное время, потраченное на решение задач, среди всех процессов-работников. $T_{теор}$ – суммарное время на решение всех задач соотнесённое к числу процессов-работников (теоретически идеальное время работы программы). $T_{пр} = T_{общ} - T_{мп}$ – потенциальный простой. $T_{дис} = T_{мп} - T_{теор}$ – максимальный дисбаланс. Самыми важными из них являются $T_{мп}$ и $T_{пр}$, которые можно использовать в качестве оценки эффективности работы программы. Идеальный результат: $T_{мп} = T_{общ} = T_{теор}$ и $T_{пр} = 0$.

**Таблица 1.** Результаты выполнения программы $N_T = 3000$, $T_w = 1$ мс.

| $N_S$ | $T_{общ}$, с. | $T_{мп}$, с. | $T_{мп} / T_{общ}$ (%) | $T_{теор}$, с. | $T_{пр}$, с. | $T_{пр} / T_{общ}$ (%) | $T_{дис}$, с. |
|---|---|---|---|---|---|---|---|
| 1 | 5,278 | 3,234 | 61 | 3,234 | 2,044 | 39 | 0 |
| 2 | 2,688 | 1,639 | 61 | 1,637 | 1,051 | 39 | 0,002 |
| 3 | 1,675 | 1,1 | 66 | 1,08 | 0,595 | 36 | 0,02 |
| 7 | 1,352 | 0,468 | 35 | 0,464 | 0,888 | 66 | 0,004 |
| 15 | 15,955 | 1,105 | 7 | 0,216 | 15,739 | 99 | 0,889 |
| 31 | 16,421 | 1,254 | 8 | 0,106 | 16,315 | 99 | 1,148 |

**Таблица 2.** Результаты выполнения программы $N_T = 3000$, $T_w = 100$ мс.

| $N_S$ | $T_{общ}$, с. | $T_{мп}$, с. | $T_{мп} / T_{общ}$ (%) | $T_{теор}$, с. | $T_{пр}$, с. | $T_{пр} / T_{общ}$ (%) | $T_{дис}$, с. |
|---|---|---|---|---|---|---|---|
| 1 | 303,86 | 300,343 | 99 | 300,343 | 3,517 | 1 | 0 |
| 2 | 152,188 | 150,247 | 99 | 150,207 | 1,981 | 1 | 0,04 |
| 3 | 101,253 | 100,164 | 99 | 100,136 | 1,117 | 1 | 0,028 |
| 7 | 43,358 | 42,958 | 99 | 42,906 | 0,452 | 1 | 0,052 |
| 15 | 20,476 | 20,034 | 98 | 20,027 | 0,449 | 2 | 0,007 |
| 31 | 9,891 | 9,716 | 98 | 9,69 | 0,201 | 2 | 0,026 |

Таблица 1 и Таблица 2 иллюстрируют зависимость основных характеристик от числа работников для различного времени выполнения задачи. Если время выполнения каждой задачи мало (Таблица 1), то система распределения нагрузки искусственно увеличивает дисбаланс. Это связано с тем, что необходимость совершать перезапуск невыполненных задач, усложняет процесс поиска свободного работника. Дисбаланс растёт, так как приоритет на получение работы оказывается неодинаковым. Например, в данной реализации, первые по номеру процессы имеют повышенный шанс на получение новой задачи. Если время выполнения одной задачи сравнительно велико (Таблица 2,Таблица ***3***), то результат стремится к идеальному. Увеличение числа процессов повышает время простоя, что негативно сказывается на эффективности работы. Но для любого числа работников можно свести время простоя к минимуму за счёт увеличения времени выполнения одной задачи. Таблица 1 демонстрирует низкую эффективность работы программы при низкой вычислительной нагрузке, но при решении реальных задач такое малое время выполнения маловероятно.

**Таблица 3.** Результаты выполнения программы $N_T = 3000$, $N_s = 6$.

| $T_w$, мс. | $T_{общ}$, с. | $T_{мп}$, с. | $T_{мп} / T_{общ}$ (%) | $T_{теор}$, с. | $T_{пр}$, с. | $T_{пр} / T_{общ}$ (%) | $T_{дис}$, с. |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1,437 | 0,547 | 38 | 0,542 | 0,895 | 62 | 0,005 |
| 5 | 2,985 | 2,556 | 86 | 2,548 | 0,437 | 15 | 0,008 |
| 10 | 5,554 | 5,065 | 91 | 5,053 | 0,501 | 9 | 0,012 |
| 15 | 8,085 | 7,565 | 94 | 7,555 | 0,53 | 7 | 0,01 |
| 20 | 10,593 | 10,08 | 95 | 10,055 | 0,538 | 5 | 0,025 |
| 25 | 13,001 | 12,558 | 97 | 12,55 | 0,451 | 3 | 0,008 |

**Таблица 4.** Результаты выполнения программы $N_S = 8$, $T_w = 1$ мс.

| $N_T$ | $T_{общ}$, с. | $T_{мп}$, с. | $T_{мп} / T_{общ}$ (%) | $T_{теор}$, с. | $T_{пр}$, с. | $T_{пр} / T_{общ}$ (%) | $T_{дис}$, с. |
|---|---|---|---|---|---|---|---|
| 1000 | 0,471 | 0,156 | 33 | 0,136 | 0,335 | 71 | 0,02 |
| 2000 | 0,946 | 0,312 | 33 | 0,271 | 0,675 | 71 | 0,041 |
| 3000 | 1,491 | 0,469 | 31 | 0,406 | 1,085 | 73 | 0,063 |
| 4000 | 2,05 | 0,626 | 31 | 0,542 | 1,508 | 74 | 0,084 |
| 5000 | 2,524 | 0,78 | 31 | 0,677 | 1,847 | 73 | 0,103 |

Таблица 3 содержит результаты исследования зависимости основных характеристик от времени выполнения одной задачи. При увеличении объёма работ уменьшается относительное время простоя. Это можно объяснить тем, что временные затраты, связанные с системой распределения работ, не зависят от сложности задач, а искусственный дисбаланс снижается. Таблица 4 показывает, что изменения числа задач не оказывает значительного влияния на относительное время простоя. Таким образом, объединяя задания в группы и высылая их одновременно, можно повысить общую эффективность работы программы.

## 4. Заключение

Проанализировав текущие подходы к обработке данных большого объёма, такие как MapReduce, Dryad и Master-Slave, с учётом основных особенностей задач сейсморазведки и опыта существующих систем обработки данных, был предложен достаточно универсальный метод построения параллельных программ. Этот подход расширяет классическую модель Master-Slave системой точек восстановления, классификацией процессов и возможностью перезапуска задач, в случае ошибок. При этом сохраняя её основные качества: масштабируемость и простой подход к балансировке нагрузки в гетерогенной вычислительной среде.

На основе предложенного метода была разработана программная реализация на языке C++ с использованием технологии MPI. Её главным недостатком является тяжеловесная система распределения работ. Но при достаточно большом времени выполнения одной задачи, сравнительное время простоя вычислительного комплекса будет мало. Анализ результатов вычислительных экспериментов позволяет сделать вывод, что предложенный метод позволяет упростить процесс построения параллельных программ на основе существующего последовательного кода, с учётом основных требований задач обработки данных сейсморазведки.

## Литература

1. Гурвич И.И., Боганик Г.Н. Сейсмическая разведка. Учебник для вузов. — 3-е изд., перераб. М.: Недра, 1980, 551 с.
2. Сухорослов О.В. Новые технологии распределенного хранения и обработки больших массивов данных // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению "Информационно-телекоммуникационные системы", 2008. - 40 с.
3. Филд А., Харрисон П. Функциональное программирование. Пер. с англ. — М.: Мир, 1993. — 637 с.
4. Dean J., Ghemawat S. MapReduce: simplified data processing on large clusters. Commun. ACM: 2008. Vol. 51, No. 1. P. 107-113.

5. Isard M., Budiu M., Yu Y., Birrell A., Fetterly D. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. European Conference on Computer Systems (EuroSys), Lisbon, Portugal, March 21-23, 2007.
6. Massingill B.L., Mattson T.G., Sanders B.A. A pattern language for parallel application programming. Technical Report CISE TR 99-022/ - University of Florida, 1999

# Parallel processing of seismic data by using the extended Master-Slave model

A.P. Burtsev[1]

Moscow State University[1]

The search of oil and gas fields is important task for modern society. The high cost of drilling and conducting of the search operations making the task of processing and analysis particularly important and relevant. Given the large amount of data collected and the high computational complexity, the usage of high-performance systems is very important for effective data processing. However, many tasks of seismic contain a wide potential for parallelization. The article will be considered the extension of the model "Master-Slave" for the organization of parallel processing of data. It takes into account the seismic problems and helps transfer the sequential algorithms on cluster architecture. This method provides: high scalability, dynamic load balancing and accounting for the specificity of heterogeneous computing clusters.

*Keywords:* Parallel computing, seismic, heterogeneous clusters, scalable parallel.

## References

1. Gurvich I.I., Boganik G.N. Seismic exploration. Textbook for students. - 3rd ed., Revised. M.: Nedra, 1980, 551 p.
2. Sukhoroslov O.V. New technologies for distributed storage and processing of large data sets // All-Russian competition overview and analytical articles on the priority direction "Information and telecommunication systems", 2008. - 40 p.
3. Field A., Harrison P. Functional programming. Trans. from English. - M.: Mir, 1993. - 637 p.
4. Dean J., Ghemawat S. MapReduce: simplified data processing on large clusters. Commun. ACM: 2008. Vol. 51, No. 1. P. 107-113.
5. Isard M., Budiu M., Yu Y., Birrell A., Fetterly D. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. European Conference on Computer Systems (EuroSys), Lisbon, Portugal, March 21-23, 2007.
6. Massingill B.L., Mattson T.G., Sanders B.A. A pattern language for parallel application programming. Technical Report CISE TR 99-022/ - University of Florida, 1999