Масштабируемые алгоритмы обеспечения отказоустойчивого счёта гибких задач на неструктурированных сетках в крупномасштабных вычислительных системах

А.Б. Новиков, Г.И. Евтушенко

ФГУП «Всероссийский научно-исследовательский институт автоматики им. Н.Л.Духова»

В работе представлен сравнительный анализ стратегий обеспечения отказоустойчивости на основе бездисковых контрольных точек. Предлагаются решения блочной декомпозиции сетки и превентивной передачи данных для обеспечения отказоустойчивого счёта. Описаны возможности предложенного алгоритма по балансировке нагрузки и добавления вычислительных ресурсов во время выполнения задач.

Ключевые слова: бездисковые контрольные точки, неструктурированные сетки, высокопроизводительные вычисления, масштабируемые алгоритмы

1. Введение

Научный прогресс в области математической физики тесно связан с возможностью эффективного использования современных вычислительных систем. Суперкомпьютерное моделирование широко применяется при решении актуальных научно-технических задач в различных высокотехнологичных отраслях промышленности, таких как атомная энергетика, авиастроение и авиационное двигателестроение, автомобилестроение и возобновляемая энергетика. Непрерывный рост производительности суперкомпьютеров открывает все более широкие возможности перед вычислительным экспериментом. Уже существуют системы, обладающие пиковой производительностью более сотен петафлопс. Несмотря на это, большая часть современных суперкомпьютеров востребована не в полной мере. Накопленный опыт применения многопроцессорных систем ориентирован, в первую очередь, на параллельные системы средней производительности, содержащие относительно небольшое число процессоров. Создание расчётных кодов, в полной мере использующих возможности современных вычислительных систем является сложной и актуальной научной проблемой [2].

Обращая внимание на то, что существующие программные комплексы, зачастую, не используют в полной мере современные крупномасштабные вычислительные системы, а также на то, что оценка необходимых вычислений на порядки превосходит возможности существующих ресурсов, можно заключить, что необходимость создания масштабируемых алгоритмов будет лишь возрастать.

При росте объёма используемых вычислительных ресурсов возникает ряд проблем, связанных с эффективностью использования вычислительного комплекса. Отказ части оборудования при крупномасштабном счёте может привести к потере результатов недельного счёта. Таким образом, отказоустойчивость программного комплекса непосредственно влияет на эффективность использования вычислительных ресурсов. Экстенсивный рост количества используемых компонент ведёт к уменьшению средней наработки на отказ (MTBF) вычислительного комплекса. Отмечается [3], что в настоящее время 20 % вычислительной мощности высокопроизводительных систем теряется по причине возникших неисправностей и восстановления после них, типичные значения MTBF составляет от 8 часов до 15 дней. Согласно прогнозам, для систем экзафлопного уровня, построенной в 2020 г., MTBF будет составлять от нескольких минут до 1 часа. При этом проявляются ограничения текуще-

го подхода - создания контрольных точек (КТ). Контрольная точка создаётся периодически. Её создание заключается в сохранений состояния счёта на надёжный носитель данных (диски). В случае сбоя, сохранённое состояние восстанавливается путём загрузки данных контрольных точек. В результатах нескольких независимых исследований [4–7] было предсказано, что потенциальные экзафлопсные вычислительные системы будут тратить более 50 % своего времени на чтение и запись КТ.

Главный источник накладных расходов во всех основанных на стабильном хранилище системах - это время, которое требуется для записи КТ в стабильное хранилище [10]. КТ на системе с десятью тысячами процессоров, подразумевает, что все важные данные для приложения на всех десяти тысячах процессоров должны быть периодично записаны на стабильное хранилище, что может ввести неприемлемое число накладных расходов в систему КТ (рис. 1).

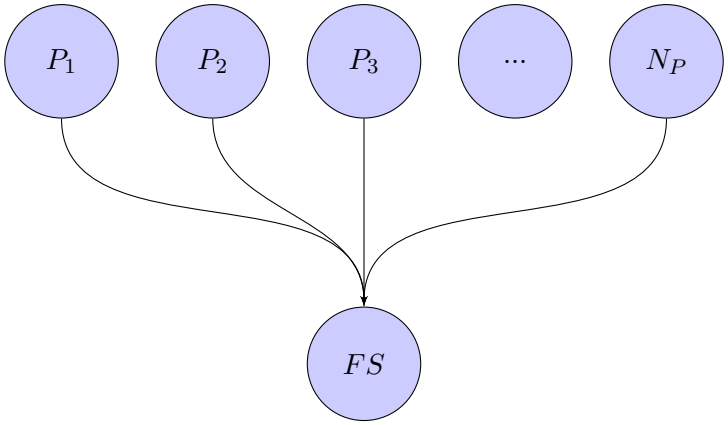

Рис. 1. Базовая стратегия «контрольная точка - рестарт»

Экспоненциально возрастающая степень параллельности требует от архитектур и программного обеспечения максимального учёта свойства локальности, обеспечивающего эффективное использование вычислительных ресурсов и минимизацию расходов на передачу данных. Только наличие приложений, способных реально использовать возможности экзафлопсных кластеров, оправдывает создание систем подобного уровня.

2. Бездисковые контрольные точки

В связи с высокой частотой сбоев и большим числом процессоров в следующем поколении вычислительных систем классический подход к отказоустойчивости может стать крайне неэффективным путём к обработке сбоев. Это предъявляет требования к исследованию масштабируемых техник к отказоустойчивости. Рассмотрим базовые подходы к созданию контрольных точек.

Контрольные точки с контролем чётности

Базовая идея контрольных точек с контролем чётности состоит в дополнении локальных контрольных точек, хранимых каждым узлом, контрольной точкой с контролем чётности (рисунок 2). Для контрольной точки с контролем чётности выделяется специальный процесс. Таким образом имеется $p_1, \ldots, p_n, p_c$ - $N+1$ узел. Пусть размер контрольной точки процесса p_i будет S_i. Процесс контрольных точек записывает каждое значение S_i, для $1 \leq i \leq n$. За S_c примем размер контрольной точки с контролем чётности, которая является наибольшим значением S_i для $i \leq i \leq n$. Пусть $b_{i,j}$ будет j-ым байтом p_i-ой контрольной точки если $j \leq S_i$ и 0 в противном случае. Каждый байт $b_{c,j}$ контрольной точки с контролем чётности является результатом побитового выполнения операции исключающего или для $1 \leq j \leq S_c$:

$$b_{c,j} = b_{1,j} \oplus b_{2,j} \oplus \ldots \oplus b_{n,j} \tag{1}$$

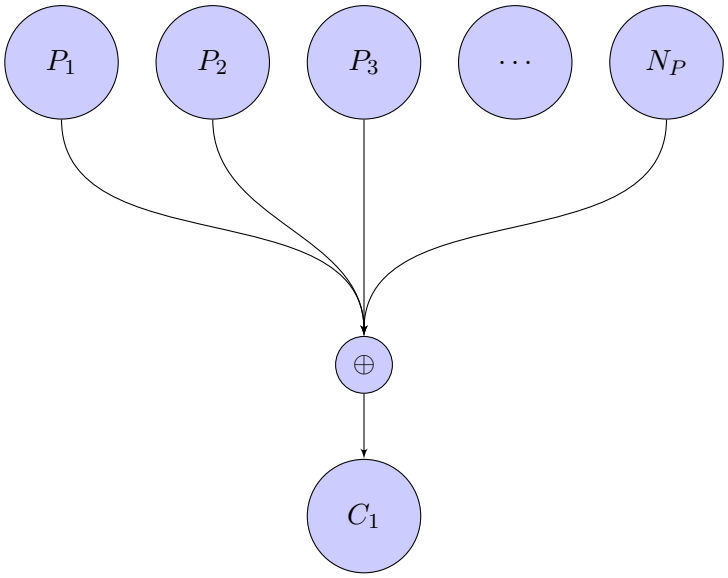

Рис. 2. Базовые бездисковые контрольные точки

Далее, данные локальной контрольной точки, хранящиеся на отказавшем узле, могут быть восстановлены вычислением для $1 \leq j \leq S_i$:

$$b_{i,j} = b_{1,j} \oplus \ldots \oplus b_{i-1,j} \oplus b_{i+1,j} \ldots b_{n,j} \ldots b_{c,j} \tag{2}$$

Этим контрольные точки напоминают технологию RAID 5. В работе [12] предлагается выделять два процесса $p_1, \ldots, p_n, p_c, p_b$ - $N + 2$, так, что процесс p_c собирает контрольные точки и выполняет над ними операции исключающего или, передавая уже не актуальные контрольные точки с контролем чётности на процессор резервного копирования p_b, хотя, безусловно, эти роли могут быть объединены.

Возможны также вариации сбора контрольных точек для создания контрольной точки с контролем чётности [10]. Наиболее простой метод - отправлять контрольные точки на процесс контрольных точек напрямую (рисунок 3).

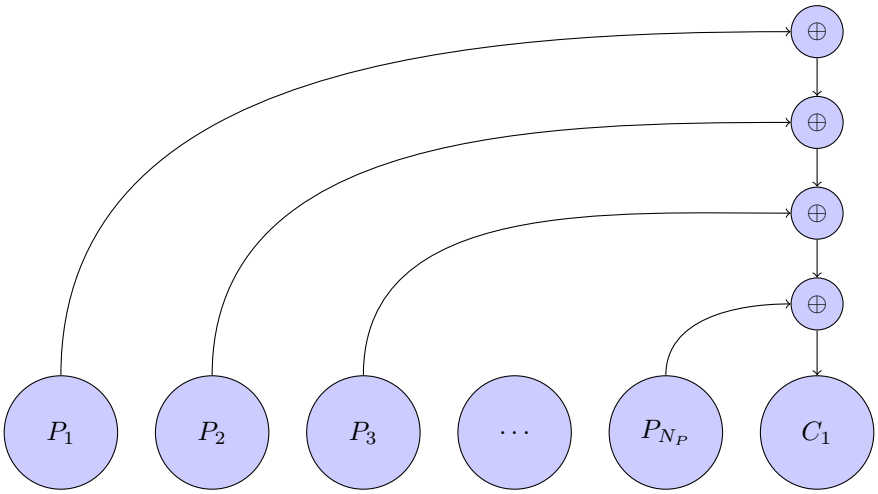

Рис. 3. Прямое кодирование контрольных точек на основе контроля чётности

При этом процесс C_1 становится узким местом системы, так как он принимает сообщения от всех процессов. Кооперация всех процессов позволяет использовать конвергентную схему кодирования контрольных точек (рисунок 4).

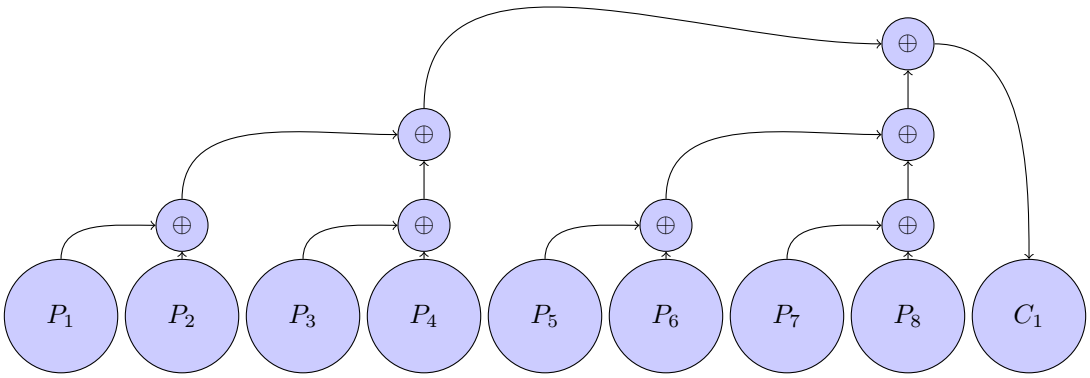

Рис. 4. Конвергентное кодирование контрольных точек на основе контроля чётности

Зеркалирование контрольных точек

Зеркалирование контрольных точек (рисунок 5) соответствует стратегии, в которой каждому процессу сопоставлен процесс контрольных точек, и i-ый процесс контрольных точек хранит контрольные точки i-го процесса. Данная стратегия позволяет устоять против сбоев n процессов, но не может быть перенесён сбой процесса контрольных точек и самого процесса. Данный тип стратегии имеет небольшие накладные вычислительные расходы, так как нет необходимости в кодировании контрольных точек.

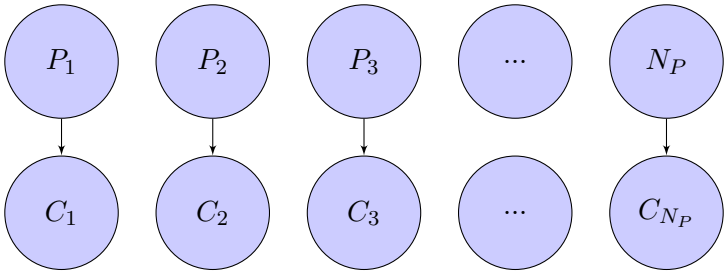

Рис. 5. Зеркалирование бездисковых контрольных точек

Одномерный контроль чётности

В одномерном контроле чётности (рисунок 6) есть $1 \leq m \leq n$ процессов контрольных точек. Процессы приложения, в свою очередь, поделены на m групп $g_1, \ldots, g_m$ приблизительно одного размера. Процесс контрольных точек i рассчитывает контроль чётности для i-ой группы. Это повышает покрытие сбоев, так как теперь может быть обработан один сбой на группу. Более того, кодирование контрольной точки теперь будет более эффективно, так как нет единственного узкого места. Одномерный контроль чётности сводится к обыкновенным контрольным точкам с контролем чётности в случае $m = 1$ и к зеркалированию контрольных точек когда $m = n$.

Двумерный контроль чётности

Двумерный контроль чётности (рисунок 7) является расширением идеи одномерного контроля чётности. Все процессы размещаются в двумерной сетке и есть процесс контрольных точек на каждую строку и столбец сетки. Двумерный контроль чётности требует $m \geq 2\sqrt{n}$ процессов контрольных точек, и может выдержать сбой любого процесса в каждой строке или столбце.

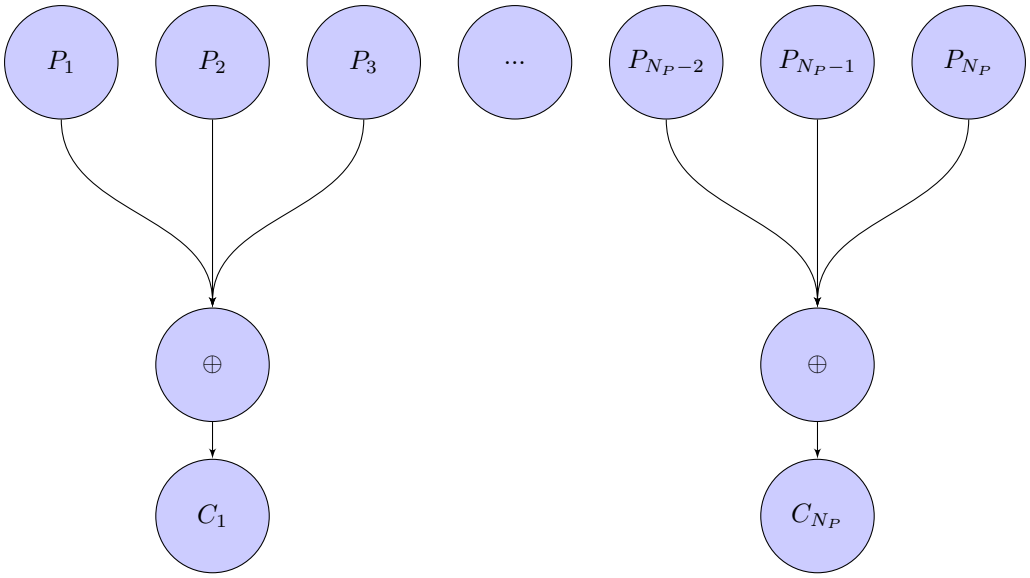

Рис. 6. Одномерный контроль чётности

Прочие схемы бездисковых контрольных точек

В контексте кодирования контрольных точек также применяют код Хэмминга, позволяющий устоять против сбоев двух произвольных процессов с грубо $\log n$ дополнительными процессами. Каждый процесс контрольных точек вычисляет контроль чётности подмножества вычислительных процессов. Также находит применение кодирование Рида-Соломона, использующее поле Галуа для кодирования контрольной точки так, что любые m сбоев могут быть обработаны с использованием m процессов контрольных точек. Так как кодирование это более сложно, чем вычисление контроля чётности, вычислительные накладные расходы превосходят прочие методы, но обеспечивают наибольшее покрытие сбоев на процесс контрольных точек [10].

Сравнение бездисковых схем обеспечения отказоустойчивости

В [10, 12] Plank предложил использовать бездисковые контрольные точки как подход к устойчивости против единичных сбоев с низкими накладными расходами в условиях недоступности стабильного хранилища. Экспериментальные исследования представленные в [16] показали, что бездисковые контрольные точки имеют намного лучшую производительность чем традиционные ориентированные на диск техники контрольных точек.

Существует несколько статей, в которых сравнивают производительность бездисковых схем контрольных точек. В [14] сравнивают производительность различных бездисковых схем контрольных точек на массивно параллельной SIMD машине. В работе реализовали три схемы контрольных точек (зеркалирование контрольных точек, контрольные точки с контролем чётности и частичные контрольные точки с контролем чётности) для приложений перемножения матриц. Их результаты показывают, что схема зеркалирования контрольных точек на порядок быстрее чем схема контрольных точек с контролем чётности, но вводит двойные накладные расходы на память. В [16] также провёл несколько экспериментальных исследований о бездисковых контрольных точках. Результаты показывают что зеркалирование контрольные точек имеет существенно лучшую производительность чем n+1 схема контрольных точек с контролем чётности. Но схема зеркалирования контрольных точек всегда представляет большие накладные расходы на память. В [10] Plank также сообщает, что схема зеркалирования контрольных точек имеет более низкие накладные расходы на производительность чем частичная схема если данные контрольной точки сохранятся на локальном диске вместо памяти процессора.

Для устойчивости произвольному числу сбоев с низкими накладными расходами на

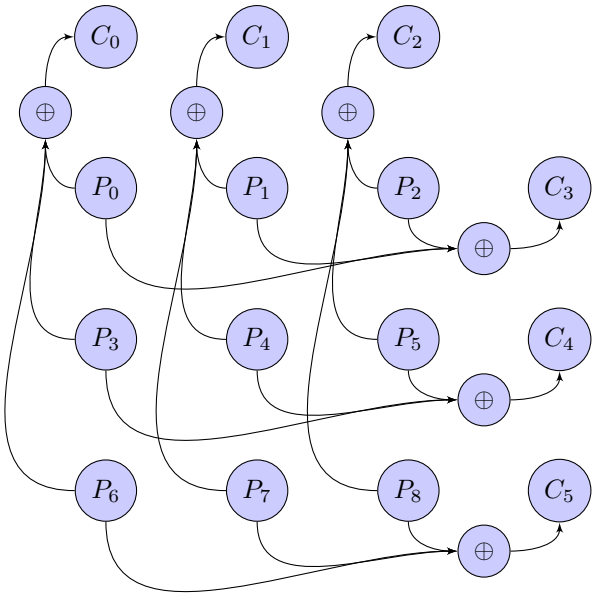

Рис. 7. Двумерный контроль чётности

производительность в [15] предложили двухуровневый подход к распределённому восстановлению. Двухуровневая схема восстановления устойчива к более вероятным сбоям с низкими накладными расходами на производительность, когда сбои с меньшей вероятностью могут быть перенесёнными с высокими накладными расходами. Так, более вероятные единичные сбои переносятся с бездисковыми контрольными точками (зеркалирование контрольных точек), когда менее вероятные многие сбои переносятся с традиционным диск-ориентированным подходом к контрольным точках. Автор демонстрирует, что для минимизации средних накладных расходов, часто важно брать как бездисковые контрольные точки, так и диск-ориентированные.

В работе [16] Silva показал, что определённые пользователем схемы контрольных точек имеют много меньшие накладные расходы чем схемы контрольных точек системного уровня. Но степень повышения производительности зависит от конкретного приложения.

В контексте моделирования отказов, в работе [17] рассмотрена возможность применения расширения ULFM стандарта MPI. В работе [18] исследована зависимость среднего времени между отказами и временем выполнения программы при использовании различных стратегий к отказоустойчивости, а также без использования последних. Показано 15% сокращение времени вычислений при использовании схемы сохранения в оперативную память по сравнению с гибридным методом сохранения контрольных точек. В работе [7] для схем работающих с лёгкими и тяжёлыми отказами и для приложений, сохраняющих контрольные точки на локальные устройства хранения, получены формулы полного времени выполнения приложения и времени уходящего на работу с отказами. В случае среднего времени между отказами менее часа, использование средств разделения на лёгкие и тяжёлые отказы позволяет сократить время вычислений более чем на 10%. При этом схема бездисковых контрольных точек была реализована так, что каждый процесс отправлял контрольную точку соседнему процессу.

При использовании перечисленных техник обеспечения отказоустойчивости возникает ряд ограничений. С одной стороны эти стратегии ограничены технологически. Наиболее распространённый стандарт построения многопроцессорных программных комплексов не предоставляет каких-либо функций связанных с обеспечением отказоустойчивости в вычислительных системах [19]. Помимо технологических ограничений, существуют ограниче-

ния масштабируемости приведённых схем. Выведение части узлов из счёта для обеспечения хранения и обработки контрольных точек может оказаться критическим с позиции производительности для схем зеркалирования или схем с одномерным или двумерным контролем чётности.

В крупномасштабных системах существуют проблема времени пересылки данных. Одним из решений является использование превентивной пересылки данных в целях сокращения числа коммуникаций. В этом контексте была разработана [20] схема $B + 2R$ (рис. 8).

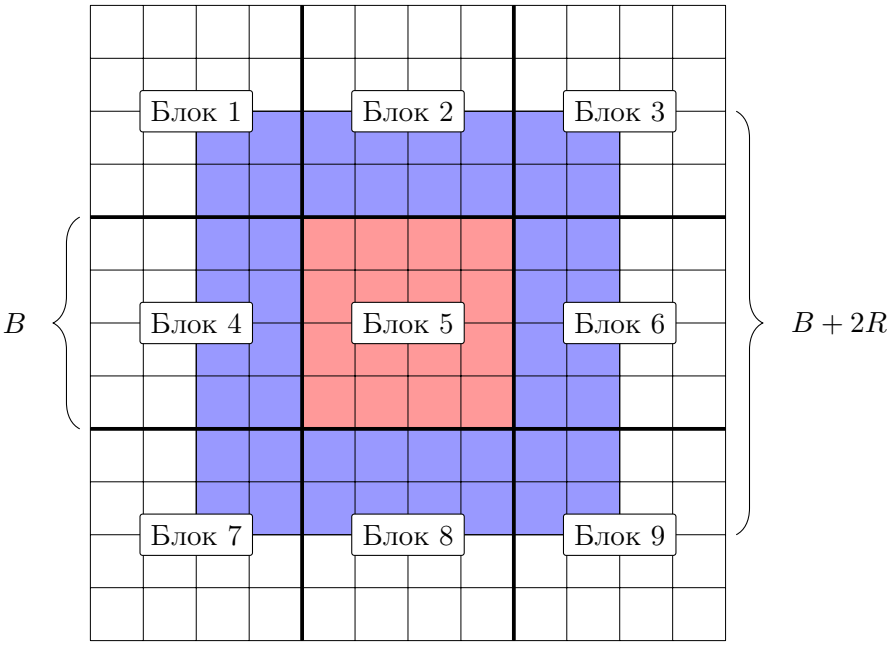

Рис. 8. Схема сокрытия латентности $B + 2R$

Данная схема позволяет производить обмен данными только раз за R итераций счёта за счёт избыточности пересылаемых данных.

Бездисковые контрольные точки, балансировка нагрузки и передача гало требуют интенсивного обмена данными. Необходимо разработать стратегию передачи данных учитывающую их перекрытие.

Модель предметной области.

Использование ячейки как атомарного элемента приводит к дополнительным накладным расходам при передаче и контроле вычислений. Предлагается архитектура вычислительного ядра, в которой атомом счёта и коммуникации является блок ячеек. Готовность блока к счёту определяется готовностью соседних блоков, а не соседними вычислительными узлами. Данное решение позволяет провести перераспределение и балансировку расчётных данных при изменении количества вычислительных ресурсов (как при восстановлении, так и динамически).

В начале расчёта происходит разделение сетки на блоки. Каждый блок может находится в трёх состояниях: готов к счёту, вычисляется, ожидает соседей. Таким образом, каждый процесс располагает тремя очередями блоков (рисунок 9).

Готовые к счёту блоки отправляются на расчёт. После окончания расчётов они попадают в очередь ожидающих и оповещают своих соседей о готовности. Если блок был последним, кого ожидает сосед, то сосед отправляется в очередь готовых к счёту. Если сосед блока расположен на другом узле, то он встаёт в очередь на отправку. Для реализации бездисковых контрольных точек и отправки гало передаются не только внешние блоки.

Для обеспечения отказоустойчивости мы предлагаем отправлять бездисковые контроль-

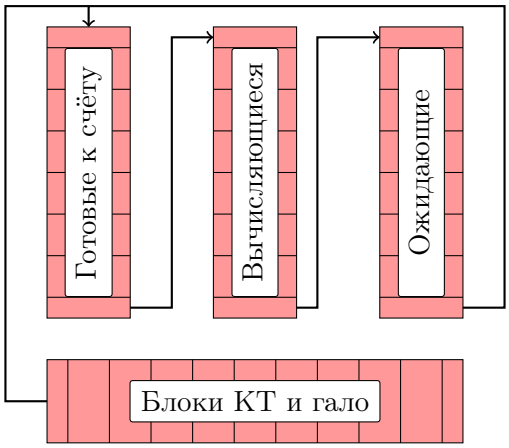

Рис. 9. Очереди блоков

ные точки на несколько соседних процессов (рисунок 10).

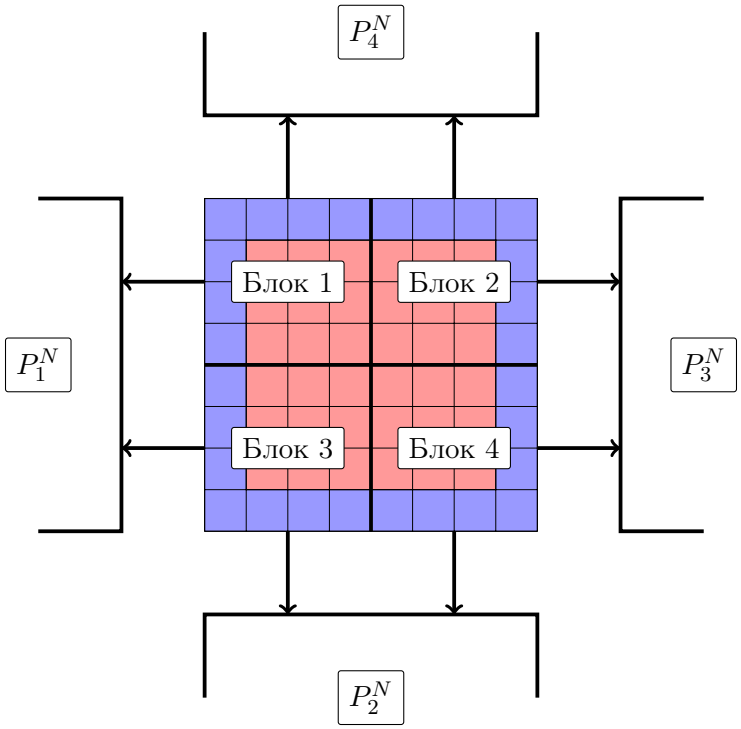

Рис. 10. Схема обмена контрольными точками

Таким образом, количество блоков, которые получит сосед P_2^N и P_4^N в простом двумерном случае, без учёта динамической балансировки нагрузки:

$$\left(\left\lfloor \frac{N_y}{2} \right\rfloor + N_y\%2\right) \cdot N_x, \tag{3}$$

где N_x количество блоков на процесс по координате x. Таким образом в худшем случае каждый процесс примет дополнительно блоков:

$$2\left[N_x\left(\left\lfloor \frac{N_y}{2} \right\rfloor + N_y\%2\right) + N_y\left(\left\lfloor \frac{N_x}{2} \right\rfloor + N_x\%2\right)\right] = 2\left[N_xN_y + N_x(N_y\%2) + N_y(N_x\%2)\right] \tag{4}$$

Из этого очевидно, что в случае чётного числа блоков по измерению (лучший случай)

мы имеем необходимость хранения блоков, занимающих дополнительно двукратный объем сетки данного процесса.

Будем обозначать количество оперативной памяти на узле как RAM. Количество ячеек в двумерном блоке обозначим как $N_x^c \cdot N_y^c$, а количество блоков на узле как $N_x^b \cdot N_y^b$. Наконец, обозначим количество процессов как N^p. Тогда количество памяти, которого не хватило на узле для хранения контрольных точек $3 \cdot size(N_x^c N_y^c N_x^b N_y^b) - RAM$. Количество процессов, которые необходимо дополнительно выделить для счёта представлено как:

$$N^{p'} - N^p = \left\lceil \frac{N^p \cdot (3 \cdot size(N_x^c N_y^c N_x^b N_y^b) - RAM)}{RAM} \right\rceil \tag{5}$$

Наихудший случай достигается, когда исходная сетка целиком занимает оперативную память процесса ($size(N_x^c N_y^c N_x^b N_y^b) = RAM$).

$$N^{p'} - N^p = \left\lceil \frac{2 \cdot N^p \cdot size(N_x^c N_y^c N_x^b N_y^b)}{RAM} \right\rceil = 2 \cdot N^p \tag{6}$$

То есть необходимо выделить дополнительно двукратное количество процессов. Стоит отметить, что на практике, в широком классе задач, оперативная память редко используется более чем на треть. Это значит, что обеспечение отказоустойчивости указанной стратегией не потребует существенного увеличения количества процессов.

В случае, когда блоки в очереди готовых к счёту закончились, имеет место ряд эвристик, на основании которых делается заключение о уведомлении удалённого процесса о присваивании его блока данному процессу, после чего происходит миграция блока между очередями и счёт продолжается без существенных задержек, в силу асинхронности коммуникации. Это обеспечивает динамическую балансировку нагрузки. Стоит отметить, что этот процесс не отягощается необходимостью передачи гало данных. Блок, который забрали у удалённого процесса становится его гало блоком, мигрируя из очереди готовых к счёту в очередь гало блоков.

При этом встаёт вопрос о поддержании когерентности информации о размещении блоков. Требование к масштабируемости всех алгоритмов создаёт ограничения использования глобальных синхронизаций. Таким образом, необходимо разработать алгоритм поддержания когерентности информации о размещении блоков опираясь лишь на коммуникации с соседними узлами. Для решения этой задачи предлагается хранить информацию о передаче блоков другим узлам. Таким образом, когда процесс запрашивает блок у процесса, на котором его уже нет, последний процесс отвечает информацией о новом (по его мнению) расположении блока. Так продолжается до тех пор, пока не будет найдено новое место хранения блока.

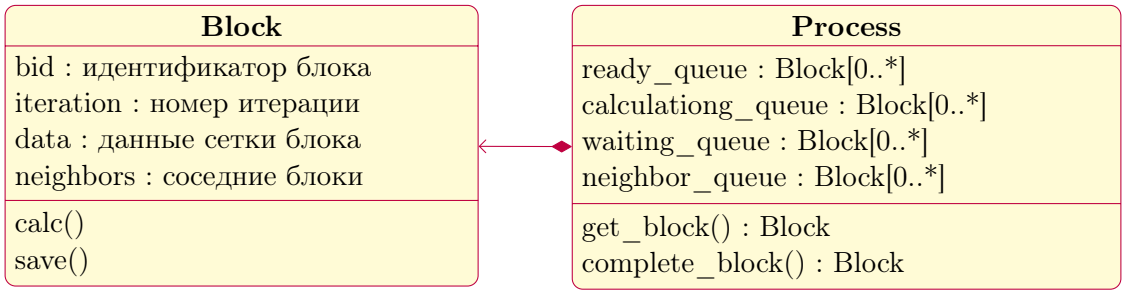

Рис. 11. Базовые классы

Ключевой в данной архитектуре оказывается структура блока, так как она должна содержать всю необходимую информацию для его счёта и учитывать возможность динамической балансировки нагрузки. В случае локального адаптирования сетки может возникнуть

ситуация, в которой один блок будет считаться на порядок дольше других блоков. Данное ограничение разрешается разделением обязанности по счёту блока между несколькими ветвями.

В общем случае функции запроса блока и его возвращения вызываются в рамках одного процесса независимо разными ветвями (рис. 11). Таким образом получается двухуровневая параллельность, на один узел выделяется один процесс.

Остаётся не раскрытым вопрос самой пересылки контрольных точек. Так как элементарной единицей коммуникации является блок, появляется возможность использования стратегии $B + 2R$ (рис. 12) для сокращения числа коммуникаций до одной в R итераций.

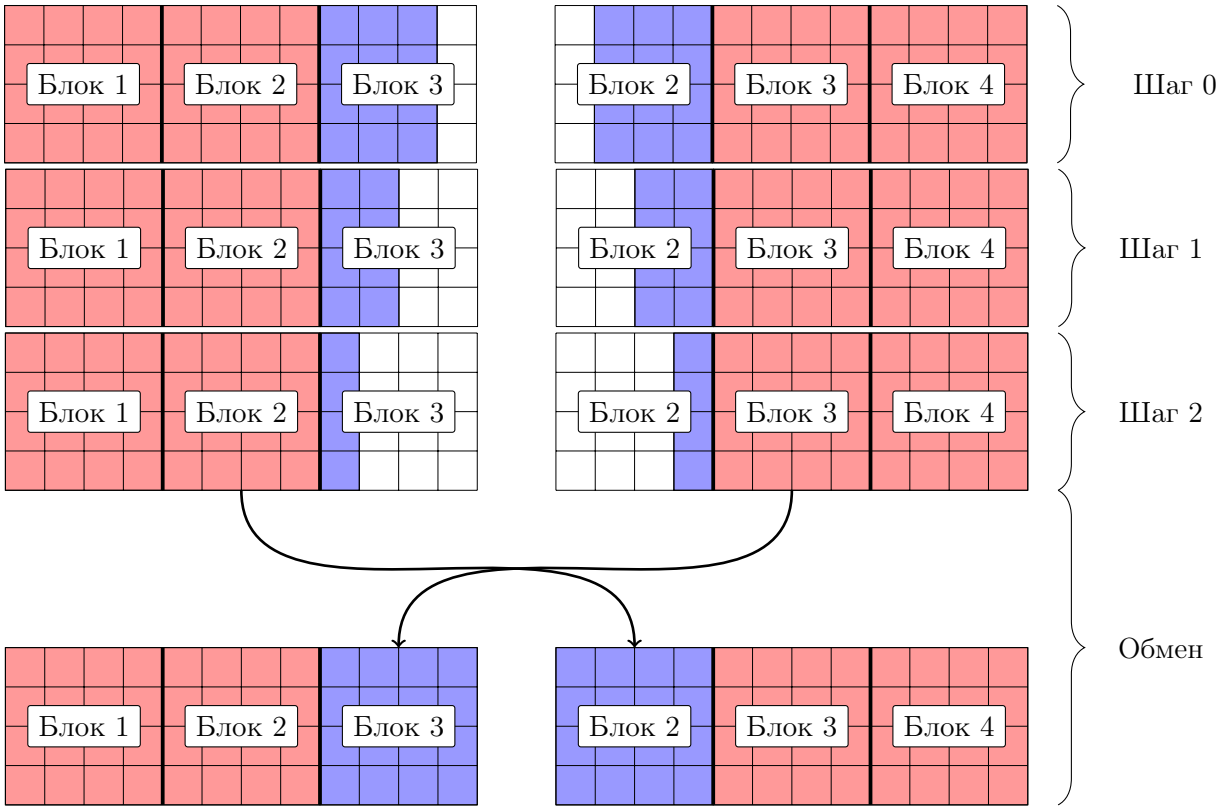

Рис. 12. Обмен гало блоками

Предполагается два варианта отправки контрольных точек. В случае нехватки оперативной памяти на узле - контрольные точки отправляются штатным образом (раз в фиксированное число итераций счёта). Второй вариант предъявляет определённые накладные расходы на оперативную память, но при этом перекрывает отправку контрольных точек вычислениями. Предполагается, что на момент создания контрольной точки делается снимок всех блоков сетки. Затем блоки из снимка начинают асинхронно отправляться на соседние узлы в соответствии с предложенной схемой (рис. 10). Таким образом получается, что отправка данных контрольных точек не привязана к итерациям счёта. На основании поведения отправки контрольных точек возможно строить механизм приоритизации и планирования пересылок контрольных точек, гало блоков (в том числе как части контрольных точек) и сохранении в центральное стабильное хранилище.

3. Заключение

Проведён обзор существующих стратегий обеспечения отказоустойчивости бездисковыми контрольными точками. Предложены масштабируемые стратегии обеспечения отказоустойчивости, динамической балансировки нагрузки и коммуникации с перекрытием дан-

ных. В основе предложенных стратегий лежит элементарный элемент счёта и коммуникации - блок. В дальнейшем предполагается определение класса задач, для которых доступно применение предложенной архитектуры вычислительного ядра.

Литература

1. Якобовский М.В. Вычислительная среда для моделирования задач механики сплошной среды на высокопроизводительных системах (Автореферат диссертации на соискание ученой степени доктора физико-математических наук), Москва, 2006.

2. Горобец А.В Параллельные технологии математического моделирования турбулентных течений на современных суперкомпьютерах (Автореферат диссертации на соискание ученой степени доктора физико-математических наук), Москва, 2015.

3. Гергель В.П., Линёв А.В., Проблемы и перспективы достижения экзафлопного уровня производительности суперкомпьютерных систем // Вестник нижегородского университета им. Н.И. Лобачевского. Нижний Новгород: Издательство Нижегородского госуниверситета, 2012

4. Elnozahy E., Plank J. Checkpointing for Petascale systems: a look into the future of practical rollback-recoveryv // IEEE Transactions on Dependable and Secure Computing . 2014. Vol. 1, Issue 2. P. 97 - 108.

5. R.A. Oldfield, S. Arunagiri, P.J. Teller, S. Seelam, M.R. Varela, R. Riesen, P.C. Rith. Modelings the impact of checkpoints on next-generation systems // 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007). 2007. P. 30 - 46.

6. B. Schroeder, G.A. Gibson. Understanging failures in petascale computers // Journal of Physics: Conference Series. 2007. Vol. 78, N. 1.

7. Бондаренко А.А., Якобовский М.В. Оптимальное сохранение контрольных точек на локальные устройства хранения // Суперкомпьютерные дни в России 2015

8. Горобец А. Масштабируемые параллельные алгоритмы повышенной точности для прямого численного моделирования задач газовой динамики и аэроакустики, 2007, 134 с.

9. Поляков С.В. Математическое моделирование с помощью многопроцессорных вычислительных систем процессов электронного транспорта в вакуумных и твердотельных микро- и наноструктурах (Автореферат диссертации на соискание ученой степени доктора физико-математических наук), 2010.

10. J. S. Plank, K. Li, M. Puening. Diskless Checkpointing // IEEE Transactions on Parallel and Distributed Systems. 1998, Vol. 9, N. 10, P. 972-986.

11. Zizhong G. Scalable techniques for fault tolerant high performance computing. 2006.

12. Plank J.S., Li. Faster Checkpointing with N+1 Parity // FTCS-24. 1994. P. 288 - 297.

13. Kim Y. Fault Tolerant Matrix Operations for Parallel and Distributed Systems. 1996.

14. Chiueh T., Deng P. Evaluation of checkpoint mechanisms for massively parallel machines // Fault Tolerant Computing, 1996. P. 370 - 379.

15. Vaidya N.H. A case for two-level recovery schemes // IEEE Transactions on Computers. 1998, Vol. 47, Issue 6. P. 656 - 666.

16. Silva L.M., Silva J.G. An Experimental Study about Diskless Checkpointing // Euromicro Conference, 1998. Proceedings. 24th, Vol. 1, P. 395 - 402.

17. Бондаренко А.А., Якобовский М.В. Моделирование отказов в высокопроизводительных вычислительных системах в рамках стандарта MPI и его расширения ULFM // Вестн. ЮУрГУ. Сер. Выч. матем. информ., 4:3 (2015), 5–12.

18. Бондаренко А.А., Подрыга В.О., Поляков С.В., Якобовский, М.В. Отказоустойчивая реализация метода молекулярной динамики на примере одного приложения // Параллельные вычислительные технологии (ПаВТ'2016)

19. Бондаренко А., Якобовский М. Обеспечение отказоустойчивости высокопроизводительных вычислений с помощью вычислений с помощью локальных контрольных точек // Вестник Южно-Уральского государственного университета. 2014, Vol. 3, N. 3.

20. Brandon G.A., Kalyan S.P., Sudip K.S. Efficient Simulation of Agent-Based Models on Multi-GPU and Multi-Core Clusters. 2010.

Scalable algorithms for fault tolerant calculation of malleable jobs on unstructured meshes in large-scale computing systems

A.B. Novikov, G.I. Evtushenko

FSUE All-Russia Research Institute of Automatics

This paper presents a comparative analysis of diskless checkpoints based fault tolerance strategies. We offer algorithms and data structures of block based mesh decomposition with preventive data transfer to provide fault tolerant calculations. The capabilities of the proposed althorithms for load balancing and computing resources addition in runtime was studied.

Keywords: diskless checkpointing, unstructured meshes, high-performance computing,scalable algorithms

References

1. Yakobovskiy M.V. Vychislitel'naya sreda dlya modelirovaniya zadach mekhaniki sploshnoy sredy na vysokoproizvoditel'nykh sistemakh [The computing environment for modeling continuum mechanics problems on high-performance systems] Moscow, 2006.

2. Gorobets A.V Parallel'nye tekhnologii matematicheskogo modelirovaniya turbulentnykh techeniy na sovremennykh superkomp'yuterakh [Parallel technology of mathematical modeling of turbulent flows on modern supercomputers] Moscow, 2015.

3. Gergel V.P., Linev A.V. Problemy i perspektivy dostizheniya ekzaflopnogo urovnya proizvoditel'nosti superkomp'yuternykh sistem [Exaflop performance of supercomputers: chalanges and trends]. Vestnik nizhegorodskogo universiteta im. N.I. Lobachevskogo [Bulletin of Lobachevsky State University of Nizhni Novgorod (UNN)], 2012

4. Elnozahy E., Plank J. Checkpointing for Petascale systems: a look into the future of practical rollback-recoveryv // IEEE Transactions on Dependable and Secure Computing . 2014. Vol. 1, Issue 2. P. 97 - 108.

5. R.A. Oldfield, S. Arunagiri, P.J. Teller, S. Seelam, M.R. Varela, R. Riesen, P.C. Rith. Modelings the impact of checkpoints on next-generation systems // 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007). 2007. P. 30 - 46.

6. B. Schroeder, G.A. Gibson. Understanging failures in petascale computers // Journal of Physics: Conference Series. 2007. Vol. 78, N. 1.

7. Bondarenko A.A., Yakobovskiy M.V. Optimal'noe sokhranenie kontrol'nykh tochek na lokal'nye ustroystva khraneniya [Optimal checkpoints saving on the local storage device] Superkomp'yuternye dni v Rossii [Russian Supercomputing Days], 2015

8. Gorobets A. Masshtabiruemye parallel'nye algoritmy povyshennoy tochnosti dlya pryamogo chislennogo modelirovaniya zadach gazovoy dinamiki i aeroakustiki [The scalable parallel algorithms for high accuracy for direct numerical simulation of gas dynamics and aeroacoustics] 2007

9. Polyakov S.V. Matematicheskoe modelirovanie s pomoshch'yu mnogoprotsessornykh vychislitel'nykh sistem protsessov elektronnogo transporta v vakuumnykh i tverdotel'nykh mikro- i nanostrukturakh [Mathematical modeling of electron transport processes in

vacuum and solid-state micro- and nanostructures using multiprocessor computer systems], 2010.

10. J. S. Plank, K. Li, M. Puening. Diskless Checkpointing // IEEE Transactions on Parallel and Distributed Systems. 1998, Vol. 9, N. 10, P. 972-986.

11. Zizhong G. Scalable techniques for fault tolerant high performance computing. 2006.

12. Plank J.S., Li. Faster Checkpointing with N+1 Parity // FTCS-24. 1994. P. 288 - 297.

13. Kim Y. Fault Tolerant Matrix Operations for Parallel and Distributed Systems. 1996.

14. Chiueh T., Deng P. Evaluation of checkpoint mechanisms for massively parallel machines // Fault Tolerant Computing, 1996. P. 370 - 379.

15. Vaidya N.H. A case for two-level recovery schemes // IEEE Transactions on Computers. 1998, Vol. 47, Issue 6. P. 656 - 666.

16. Silva L.M., Silva J.G. An Experimental Study about Diskless Checkpointing // Euromicro Conference, 1998. Proceedings. 24th, Vol. 1, P. 395 - 402.

17. Bondarenko A.A., Yakobovskiy M.V. Modelirovanie otkazov v vysokoproizvoditel'nykh vychislitel'nykh sistemakh v ramkakh standarta MPI i ego rasshireniya ULFM [Simulation of failures in high-performance computing systems under mpi-ulfm] Vestn. YuUrGU. Ser. Vych. matem. inform [Bulletin of the South Ural State University], 2015, vol. 4, no. 4, P. 5-12.

18. Bondarenko A.A., Podryga V.O., Polyakov S.V., Yakobovskiy, M.V. Otkazoustoychivaya realizatsiya metoda molekulyarnoy dinamiki na primere odnogo prilozheniya [The fault-tolerant implementation of the method of molecular dynamics on the example of one application] Parallelnye vychislitelnye tekhnologii (PaVT'2016) [Parallel Computational Technologies (PCT'2016)]

19. Bondarenko A., Yakobovskiy M. Obespechenie otkazoustoychivosti vysokoproizvoditel'nykh vychisleniy s pomoshch'yu vychisleniy s pomoshch'yu lokal'nykh kontrol'nykh tochek [Fault Tolerance for HPC by Using Local Checkpoints] Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta [Bulletin of the South Ural State University] 2014, Vol. 3, no. 3, P. 20-36.

20. Brandon G.A., Kalyan S.P., Sudip K.S. Efficient Simulation of Agent-Based Models on Multi-GPU and Multi-Core Clusters. 2010.