УДК 004.75

DOI: http://doi.org/10.25728/pu.2020.4.7

КОМПЬЮТЕРНЫЕ КЛАСТЕРЫ С БЫСТРЫМ АППАРАТНЫМ ВЫПОЛНЕНИЕМ СИНХРОНИЗАЦИИ СООБЩЕНИЙ И РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ СЕТЕВЫМИ СРЕДСТВАМИ

Г.Г. Стецюра

Аннотация. Предложены сетевая структура и методы быстрого взаимодействия компьютеров в новом виде распределенного составного кластера. Составной кластер организован иерархически и состоит из группы простых кластеров, один из которых выдает задания простым кластерам. Простые кластеры выполняют задания синхронно и асинхронно. В простом кластере группа процессоров также действует синхронно или асинхронно, применяя быструю барьерную синхронизацию. Действиями компьютеров простого кластера управляет ведущий компьютер — лидер. Отмечено, что в составном кластере быстро выполняются: процессы синхронизации передаваемых сообщений, процессы разрешения конфликтов доступа компьютеров к сетевым инструментам, распределенные логические операции, распределенное определение *max* и *min*, распределенные операции сложения и вычитания. Показано, что эти операции не требуют задержки сообщений для выполнения, продолжительность операций не зависит от числа компьютеров кластера, участвующих в них. Для этого компьютеры отправляют сообщения одновременно, создавая групповое сообщение, в котором одноименные биты объединяются во времени. Ускорение указанных выше распределенных вычислений и синхронизации достигается при интенсивном обращении компьютеров к сети кластера, что отличает предлагаемые решения от сложившейся практики применения компьютерной сети. Указано, что предложенные операции позволяют создавать более быстрые алгоритмы для решения задач реального времени, в том числе для задач управления работой кластера.

Ключевые слова: компьютерный кластер, иерархическая структура связей, быстрые вычисления в сети, динамическая реконфигурация, распределенная синхронизация, барьерная синхронизация, распределенный ускоритель вычислений.

ВВЕДЕНИЕ

Возможности распределенных компьютерных систем, в которых группа компьютеров объединяется сетью, в значительной степени зависят от возможностей сети. Традиционно всю обработку данных и управление работой системы выполняют компьютеры системы, а сеть занимается только передачей сообщений.

Однако в последние два десятилетия компьютерные сети существенно усложнились и начинают дополнительно выполнять различные виды обработки данных в процессе их передачи. Появились широко известные активные сети (Active Networks) [1], программно-управляемые сети (SDN) [2], сети с внутрисетевыми вычислениями (In-Network Computing) [3—6]. Такие, конструктивно достаточно сложные сети, содержат сетевые компьютеры, программное обеспечение которых выполняет дополнительно часть работы основных компьютеров. В Active Networks выполняется управление пересылкой через сеть сообщений с учетом их особенностей. Сети SDN динамически реконфигурируют сетевую структуру. В In-Network Computing сетевые средства выполняют вычисления над передаваемыми в сообщениях данными. Перечисленные сети ориентированы на передачу сообщений большого объема. Благодаря передаче части работы основных компьютеров системы программам сетевых компьютеров эффективность системы повышается.

В последние годы в Институте проблем управления им. Трапезникова РАН выполнен цикл исследований по передаче сетевым средствам распределенных компьютерных систем ряда вычислительных и управляющих операций. По сравнению с упомянутыми системами ставились несколько отличающиеся цели. Сети должны эффективно работать с короткими сообщениями. Сетевые средства должны быть простыми, не содержать компьютеры. При этом они должны выполнять распределенные вычислительные и управляющие операции в сетевых средствах с высокими скоростями, близкими к достижимым в компьютере, т. е. система с сетевыми связями должна работать как распределенный компьютер.

Удовлетворяющие этим требованиям решения были получены для суперкомпьютеров [7, 8] и систем мобильных объектов, в частности роботов [9]. Но в этих работах не рассмотрен распространенный вид систем — составные компьютерные кластеры, которые содержат простые кластеры, взаимно удаленные на расстояние не более нескольких десятков метров. Настоящая статья дает для составных кластеров способы достижения указанных целей.

В составном кластере группы простых кластеров объединяются иерархически организованной локальной компьютерной сетью. Каждый простой кластер состоит из компьютеров, объединенных между собой через простое устройство — модуль связи, который не коммутирует связи. Его задачи — выполнение распределенных вычислений и ретрансляция сигналов. Кластерами в иерархической структуре управляет один из кластеров. Сетевые средства составного кластера не содержат сложных программируемых устройств.

Над содержимым групповых сообщений быстро выполняются вычисления в модулях связи простых кластеров. Вычисления выполняются в процессе передачи сообщений, без их задержки. Время выполнения рассмотренных далее распределенных вычислений не зависит от числа участвующих в них компьютеров.

Структура изложения материала: в § 3 — центральном в статье — описывается структура и функционирование составного, иерархически организованного кластера, объединяющего модулями связи простые кластеры, описанные в § 1, в сложную структуру. В § 2 описываются выполняемые в кластерах высокоскоростные процессы синхронизации, асинхронной барьерной синхронизации, разрешения конфликтов доступа. В § 4 даны примеры быстрых распределенных вычислений в предлагаемых кластерах.

В статье использованы результаты организации распределенных быстрых вычислений в суперкомпьютерах и в группах мобильных роботов из публикаций [7—9], расширенные и адаптированные к потребностям составного компьютерного кластера с иерархическими сетевыми соединениями.

1. ПРОСТОЙ КЛАСТЕР

В простом кластере (рис. 1) применяются два вида объектов (сетевых узлов) — сетевые контроллеры *NC* и модули связи *MS*.

Объект *NC*, имеющийся у каждого компьютера кластера, участвует во всех распределенных операциях, необходимых компьютеру. Если не возникает неоднозначности, то будем называть объектом *NC* компьютер и его контроллер.

В простом кластере имеется только один объект *MS*. Объекты *NC* соединены с модулем связи *MS* каналами связи (оптоволоконными, беспроводными оптическими каналами или радиоканалами). В структуре (рис. 1, *а*) применяется любой из указанных каналов.

Объект *NC* взаимодействуют через модуль связи *MS* следующим образом. Объект *NC* должен отправлять сигналы частотой f_1 для передачи двоичной единицы и частотой f_0 для передачи двоичного нуля в модуль связи *MS,* который преобразует эти сигналы в сигналы других частот ${}^*f_1$ и ${}^*f_0$ и отправляет их всем объектам *NC*. Эти сигналы имеют одинаковую длительность для всего кластера.

Для управления асинхронными процессами с помощью барьерной синхронизации используются сигналы f_2 и ${}^*f_2$. Сигналы f_2 отправляют объекту *NC*. Каждый сигнал f_2 имеет длительность, которая требуется объекту *NC* для выполнения его части асинхронного процесса. Модуль *MS* отправляет всем *NC* сигналы ${}^*f_2$, если *MS* принимает хотя бы один сигнал f_2.

Для высокоскоростных операций, которые описаны далее, необходимо измерить время передачи сигнала между *MS* и каждым *NC*. Так как компьютеры кластера стационарны, то эти временные интервалы определяют один раз. Если не требуются выполнять распределенные операции очень быстро, то эти интервалы можно не измерять.

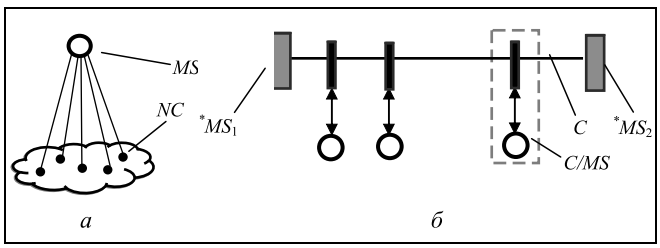

Рис. 1. Виды соединений в кластере: *а* — канал любого вида; *б* — оптоволоконный канал

В структуре (рис. 1, *б*) применяется только оптоволоконный канал. Здесь *NC* и *MS* соединены каналом, обозначенным как *C*. Эта структура позволяет кластеру иметь группу модулей *MS* и быстро заменять текущий модуль любым другим модулем *MS*. В составном кластере эта структура также позволяет составляющим его простым кластерам быстро обмениваться информацией. Для упрощения технических средств достаточно ограничиться одним или двумя модулями *MS*, расположенными на концах кабеля ($^*MS_1$ и $^*MS_2$ на рис. 1, *б*). Два модуля повышают отказоустойчивость структуры. Всегда работает только один модуль, быстро заменяемый другим при отказе первого. В этой структуре функции модуля *MS* может выполнять не только специально выделенный объект *MS*, но и любой объект *NC*.

Как показано, модуль связи *MS* — единственный вид сетевого устройства, через которое проходят все сетевые потоки. В § 2 и 4 показано, что процессы синхронизации и распределенных вычислений в значительной степени выполняются в модуле *MS*, который не содержит компьютеры, для большинства операций в нем не требуются даже логические элементы. В § 3 для работы в иерархической сетевой структуре модули *MS* составного кластера усложнены и в них применяется аппаратное управление переключением проходящих через *MS* потоков данных и команд.

2. СИНХРОНИЗАЦИЯ В ПРОСТОМ КЛАСТЕРЕ, ФОРМИРОВАНИЕ КЛАСТЕРА

Рассмотрим три вида синхронизации: синхронизацию, устранение конфликтов доступа к *MS* и барьерную синхронизацию.

Синхронизация обеспечит доставку сообщений группы *NC* в *MS* в виде одного сообщения. В этом сообщении биты с одинаковыми именами в сообщениях *NC* перекрываются в *MS* или сообщения передаются одно за другим без временных пауз, как единое сообщение [7—9].

Синхронизация с помощью специальной команды начала синхронизации. Пусть такая команда поступила от *MS* ко всем *NC*. Пусть известно T_i — время распространения сигнала между NC_i и *MS*, и $T_{\max}$ — время, не меньше времени распространения сигнала между *MS* и самыми удаленными от него *NC*.

Один из объектов *NC* отправляет команду начала синхронизации через *MS* всем *NC*. Получив команду, *NC*, которые должны отправить сообщение, отправляют в *MS* сообщение — двоичную шкалу. Число битов в шкале равно числу всех *NC*. Объекты *NC*, которые должны отправить сообщение, отправляют единицу (f_1) в позицию (бит) шкалы, которая соответствует *NC*. Шкала отправляется с задержкой $D_i = 2(T_{\max} - T_i)$. Одноименные биты в шкале поступают в *MS* одновременно от всех *NC*, и все *NC* получают общую шкалу от *MS* — результат перекрытия сигналов *NC*. Получив шкалу, *NC* отправляет сообщение с задержкой D_i. Сообщения передаются одно за другим или с перекрытием их разрядов. Все *NC* получают такое сообщение от *MS*.

Синхронизация без специальной команды. Если *NC* не получают в течение заданного интервала времени сигналы от *MS*, то они начинают передавать сообщения. Если при этом возникает конфликт, то в сообщениях *NC* в результате наложения в *MS* сигналы искажаются (совместный приход в *NC* сигналов f_1 и f_0). Первое такое искажение *NC* воспринимают как команду начала синхронизации и устранения конфликта. В ответ *NC* посылают в *MS* приведенную выше шкалу. Далее выполняется синхронизация, как описано выше, и конфликты устраняются для всех *NC* одновременно.

Барьерная синхронизация в кластере следит за моментом завершения всех асинхронных процессов, выполняемых компьютерами кластера. Обычно она выполняется программными средствами и требует много времени. В предлагаемом варианте асинхронная барьерная синхронизация выполняется аппаратно. Объекты *NC*, начав асинхронное задание, посылают сигналы в *MS* на частоте f_2. Модуль *MS* возвращает всем *NC* сигнал частоты $^*f_2$. По завершении задания каждый *NC* удаляет сигнал f_2. Когда в *MS* исчезает сигнал f_2, передача $^*f_2$ прекращается, что служит моментом синхронизации *NC*.

Таким образом, в отличие от приведенных во Введении систем, длительность каждого из рассмотренных видов синхронизации зависит главным образом от длительности переноса сигналов между *NC* системы. Использование задержки D_i делает время переноса сигнала между любым объектом и *MS* одинаковым и равным $T_{\max} = cL$, где c — скорость света, L — расстояние между *MS* и наиболее удаленными от *MS* объектами. Все виды синхронизации выполняются аппаратно. При участии в процессе синхронизации n объектов достигается высокая скорость процесса благодаря тому, что их специальные сообщения (шкалы) передаются в виде одного сообщения с совмещением в *MS* одноименных разрядов сообщений. При этом конфликт устраняется одновременно для всей группы объектов. Время синхронизации уменьшается в n раз, где n — число объектов.

Важно отметить, высокая скорость вычислительных операций из § 4 также достигается благодаря применению шкал.

3. СОСТАВНЫЕ КЛАСТЕРЫ

Составной кластер содержит центральный простой кластер нулевого уровня и подчиненные ему простые кластеры первого уровня, в которых группы компьютеров не находятся в прямой связи с модулем связи *MS* центрального кластера. Далее представлены синхронные и асинхронные составные кластеры.

3.1. Синхронный составной кластер

Структура составного кластера (рис. 2) одинакова для синхронного и асинхронного кластеров. Здесь *MS* — основной модуль связи нулевого уровня, принадлежащий центральному кластеру. Объекты *NC* нулевого уровня напрямую связаны с *MS*. В подчиненных кластерах объекты *NC* подключены к *MS* через свои модули связи первого уровня ${}^1MS$. Волоконно-оптические каналы соединяют модули первого уровня ${}^1MS$ с модулем *MS*. Удаленность каждого *NC* составного кластера от *MS* и наибольшая удаленность всех *NC* от *MS* известны.

Модуль *MS* работает, как описано выше. Модули ${}^1MS$ принимают сигналы f_0, f_1 и f_2 от подключенных к ним *NC* и, не изменяя частоту сигналов, передают их в *MS*. Поэтому *MS* воспринимает модули ${}^1MS$ как *NC*. Модуль *MS* посылает сигналы ${}^*f_1$, ${}^*f_0$, и ${}^*f_2$ не только своим *NC*, но и модулям ${}^1MS$. Последние, подобно *MS*, посылают эти сигналы своим *NC*. Модули ${}^1MS$ не преобразуют проходящие через них сигналы. Система с модулями ${}^1MS$ ведет себя аналогично системе, которая содержит только модули *MS*.

Процесс синхронизации в составном кластере имеет свои особенности. Для упрощения предположим, что кластеры пользуются шкалами синхронизации, имеющими одинаковое число двоичных разрядов.

В процессе синхронизации участвуют все простые кластеры, которые передают в сообщениях общую шкалу. От *MS* общая шкала возвращается всем *NC*. После синхронизации *NC* отправят свои сообщения без конфликтов.

В этом процессе должно выполняться дополнительное условие: сообщения *NC* должны поступать без конфликтов не только в *MS*, но и в ${}^1MS$. Каждый *NC* должен отправить сообщение с рассмотренной выше задержкой T_i, к которой добавляется задержка, учитывающая расстояние между *MS* и

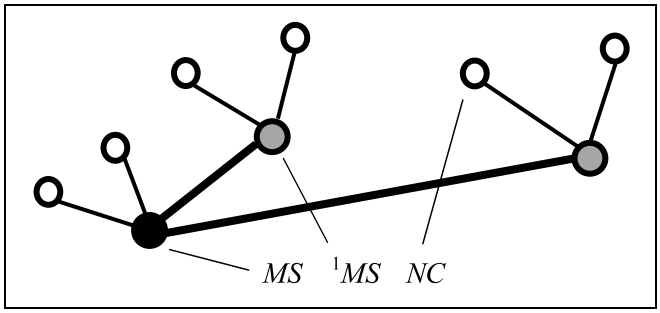

Рис. 2. Составной кластер

${}^1MS$. Все последующие операции в кластере начинаются одновременно.

В синхронном составном кластере барьерная синхронизация, цифровые операции и методы обмена данными аналогичны решениям простого кластера.

Кроме того, легко работать со шкалами и сообщениями различной длины, если длины известны заранее. Допустимы динамически изменяемые длины сообщений; однако эти длины должны быть указаны в шкале синхронизации. Для ускорения можно применять две шкалы — шкалу синхронизации, за которой следует более короткая шкала, отправляемая только *NC*, передающими сообщения. Во второй шкале указываются длины сообщений.

3.2. Асинхронный составной кластер

В отличие от п. 3.1, *NC*, входящие в любой простой кластер, работают в асинхронном составном кластере автономно, не задерживая работу *NC* других простых кластеров.

Вернемся к барьерной синхронизации в простом кластере. Предположим, один из объектов *NC*, объект *U*, отправляет задание другим объектам *NC*. Каждый из этих *NC* начинает асинхронную операцию, отправляет сигнал f_2 барьерной синхронизации в *MS* и завершает передачу сигнала f_2 после завершения задания объекта *U*. Модуль *MS* в ответ на сигнал f_2 отправляет сигнал ${}^*f_2$ объектам *NC* и завершает передачу сигнала ${}^*f_2$ в отсутствие сигнала f_2. После этого объект *U* отправляет новое задание.

Теперь обратимся к составному асинхронному кластеру. Модули ${}^1MS$ состоят из двух частей: ${}^1MS_a$ и ${}^1MS_b$. Для автономной работы *NC* в модулях ${}^1MS$ потребуются два источника сигнала f_2.

Перевод центральным кластером модуля ${}^1MS$ в автономный режим состоит из следующих шагов.

Шаг 1. Произвольный *NC* центрального кластера, действующий как *U*, отправляет команду *C*1.

После получения команды *C*1 модуль ^1MS информирует подключенные к нему *NC* о предстоящей автономной работе. Модули ^1MS и их *NC*, получившие команду *C*1, ждут команду *C*2 от объекта *U*. Эта команда перечисляет имена *NC* — получателей *C*2. Для каждого *NC* указывается задача — команда или программа, которую должен выполнить *NC*. Затем модуль ^1MS выполняет шаг 2.

Шаг 2. После получения *C*2 модуль ^1MS запрещает прохождение сигналов от *MS* к *NC*, указанного модуля ^1MS и сигналов этих *NC* к *MS*. Часть ^1MS_a отправляет в *MS* сигнал f_2 — признак барьерной синхронизации. От модуля *MS* поступает сигнал $^*f_2$. ко всем ^1MS. На шаге 3 объекты *NC* модуля ^1MS будут асинхронно выполнять задачи, указанные в команде *C*2.

Шаг 3. Команда *C*2 среди *NC* модуля ^1MS выбирает произвольный *NC*, который обозначим $^*U$. Объект $^*U$ должен управлять действиями остальных объектов *NC* модуля ^1MS. Объект $^*U$ также может быть выбран совместными операциями объектов *NC*, объединенных модулем ^1MS. В своем простом кластере объект $^*U$ ведет себя подобно объекту *U* центрального кластера.

Каждый *NC*, получивший команду от $^*U$, отправляет части ^1MS_b сигнал f_2. Часть ^1MS_b преобразует сигнал f_2 в сигнал $^*f_2$, который отправляется объектам только *NC* этого модуля ^1MS.

После завершения операции *NC* отключает свой сигнал f_2. При отсутствии сигналов f_2 модуль ^1MS_b отключает сигнал $^*f_2$.

В ответ объект $^*U$ либо продолжает работать с объектами ^1MS, либо завершает работу и отправляет команду части ^1MS_a, чтобы прекратить отправку сигнала f_2 в модуль *MS*.

Шаг 4. Когда модуль *MS* прекратит прием сигнала f_2, модуль *MS* прекратит передачу сигнала $^*f_2$ своим объектам *NC* и модулям ^1MS.

После выполнения шагов 1—4 результаты вычислений в модулях ^1MS остаются в тех *NC*, которые подключены к этим модулям, и недоступны *NC*, которые подключены к *MS*. Чтобы этим *NC* дать доступ, выполняется шаг 5.

Шаг 5. Предполагается, что в команде *C*2 объект $^*U$ получил задачу по сбору данных из *NC*, которые подключены к модулю ^1MS, и собрал требуемые данные.

После этого объект *U* получает через модуль *MS* и $^*U$ все данные из объектов.

Таким образом, разделение модуля ^1MS на части ^1MS_a и ^1MS_b позволяет взаимодействовать двум асинхронным процессам. Часть ^1MS_a обеспечивает подключение модуля ^1MS как объекта к *MS*. Часть ^1MS_b позволяет объектам, подключенным к модулям ^1MS, работать автономно. По сравнению с *MS* подчиненные модули ^1MS пришлось усложнить.

Шаги 1—5 легко изменяются для включения модулей следующих уровней иерархии — ^2MS, ^3MS и т. д. Это позволяет создать многоуровневую структуру.

В составном кластере любой компьютер можно заменить составным кластером.

Организация составного кластера с иерархической сетевой структурой связей — центральная задача статьи. Иерархическая сетевая структура характерна для многих современных распределенных компьютерных систем. Но рассмотренный выше составной кластер при наличии в нем *n* компьютеров *n*-кратно ускоряет синхронизацию и ряд видов распределенных вычислений.

3.3. Быстрые изменения в структуре кластера

Рассмотрим типичные варианты изменений в структуре кластера.

Смена лидера. Для проведения в кластере вычислений (§ 4) часто нужен объект *NC* — лидер, управляющий действиями других объектов *NC* кластера. Если лидер перестает работать, требуется выбор другого лидера за минимально возможное время. Для этого достаточно выполнить следующие действия.

Объектам *NC* присваивают различные двоичные номера. Цифра номера, равная единице, будет передаваться сигналом f_1, а цифра ноль — сигналом f_0. Выполняется алгоритм из двух шагов.

Шаг 1. Если в течение времени $2^*T_{\max}$ объекты *NC* не обнаруживают сигнал f_0 или f_1, то они переходят к шагу 2.

Если объект обнаруживает сигнал f_1, и его старшая цифра равна нулю, тогда он прекращает выполнение алгоритма. В противном случае он переходит к шагу 2.

Шаг 2. Объекты *NC* передают старшую цифру своего номера, которую получают все *NC* и ожидают в течение времени $2^*T_{\max}$. Если при этом *NC*, который передал сигнал f_0, получает сигнал f_1, то он прекращает выполнение алгоритма. Иначе *NC* повторяет шаг 2, передавая следующую цифру своего номера. Алгоритм завершается после передачи объектами *NC* последней цифры.

После завершения алгоритма будет выделен единственный *NC*. Он становится лидером или передает функции лидера другому *NC*. Число повторений шага 2 логарифмически зависит от числа объектов *NC*. Переход к представлению чисел, показанному в § 4, дает дальнейшее ускорение благодаря уменьшению числа обменов сообщениями в сети. Более сложные действия по выбору лидера изложены в работе [9].

Смена модуля связи. До сих пор мы рассматривали случай, когда модуль *MS* действует как центр, что делает кластер зависимым от его работоспособности. Снимем это ограничение.

Вначале соединение между модулями связи будем считать беспроводным (см. рис. 1, *а*). Рассмотрим общий случай — сбой или отказ *MS* или первоначальный запуск еще не сформированного кластера. Будем считать, что имеется несколько *MS*, один из них работает в данный момент времени. Остальные следят за его работоспособностью и при отказе проводят выбор нового *MS*, подобно выбору лидера среди *NC*.

При начальном запуске лидер после выполнения алгоритма требует от всех включенных в кластер *NC* определить T_i и $T_{\max}$. Существенно, что начальный запуск создает кластер полностью децентрализовано.

Если соединения оптоволоконные, то применяется структура соединений, представленная на рис. 1, *б*. Работающий и резервные *MS* объединены каналом связи согласно рис. 1, *б*. Выбор нового *MS* выполняется, как в предыдущем варианте для рис. 1, *а*, но с посылкой сигналов в соответствии с § 1.

Глобальный обмен сообщениями. Полученные в каком-либо *NC* данные в составном кластере непосредственно доступны *NC*, соединенными с тем же модулем связи. Другие *NC* получат доступ к этим данным только через указанную в п. 3.1 и 3.2 цепочку переключений в структуре *MS*. Для дальнейшего ускорения вычислений, воспользуемся объединением всех *NC* дополнительной связью в соответствии с рис. 1, *а* или 1, *б*. Эта связь включается по команде основного модуля связи нулевого уровня. В результате составной кластер преобразуется в простой кластер, действия которого описаны выше.

3.4. Управление группой распределенных конвейерных вычислений

В обычных конвейерных системах компьютеры соединены в цепочку, через которую продвигаются обрабатываемые данные. Каждый следующий компьютер цепочки использует результаты вычислений предыдущего компьютера. В предлагаемом кластере применена иная организация одновременно действующей группы конвейеров. В ней на каждом шаге процесса любому компьютеру доступна информация, адресованная любому компьютеру любого кластера, и допустимо изменять состав конвейеров.

Конвейерные вычисления начинаются с отправки команды *Ср* всем объектам для предварительной настройки конвейера. Команда *Ср* укажет каждому объекту его конвейер, место в цепочке конвейера и начальные данные для запуска процесса. После получения команды *Ср* объекты в указанном порядке выполняют этапы конвейера и передают результаты расчета в *MS*. Все объекты получают эти результаты от *MS* и используют их в расчетах следующих шагов.

Возможно синхронное и асинхронное продвижения конвейера. При синхронном процессе продолжительность отдельных шагов может различаться, однако они известны всем компьютерам кластера. Переход к следующему шагу не требует отправки дополнительных синхронизирующих сигналов. В асинхронном конвейере применяется барьерная синхронизация (см. § 2). Каждый шаг начинается после завершения предыдущего шага, с исчезновением получаемого от *MS* сигнала ${}^*f_2$.

Вся необходимая для очередного шага информация от всех участников передается синхронно в одном сообщении, что существенно ускоряет настройку шага процесса. Изложенные в § 4 распределенные операции используют короткое сообщение с объединением разрядов сообщений источников. Все приемники сообщений получают сообщение группы источников одновременно.

Приведенное ускорение требует применения синхронизации, описанной в § 2.

В публикациях, модифицирующих сетевые взаимодействия в суперкомпьютерах [7, 8], дано более гибкое и быстрое решение, но существенно более сложное в технической реализации.

4. РАСПРЕДЕЛЕННЫЕ СЕТЕВЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ОПЕРАЦИИ

Далее приведены примеры быстрых распределенных вычислительных операций, разработанных ранее для систем из публикаций [7—10] и настоящей статьи. Состав операций расширяется при переходе к новым видам и применениям систем. Поэтому § 4 служит ориентиром для создания новых полезных сетевых операций.

4.1. Формат данных в выполняемых кластером распределенных операциях

Для представления двоичных единиц и нулей применяются соответственно сигналы двух частот f_1 и f_0 (см. § 1).

В ряде операций для представления цифр применяется также двоичная шкала, в которой число битов равно основанию данной системы счисления. Только один бит в шкале, соответствующий значению цифры, равен 1, остальные равны 0. Например, для десятичной системы счисления и цифры 7 шкала равна 001000000. Для двоичной системы мы получаем обычное двоичное представление чисел. Увеличение базы системы счисления ускоряет работу, поскольку уменьшает количество обменов сообщениями.

4.2. Распределенные цифровые операции

Битовые логические операции *И* и *ИЛИ*. Объекты *NC* синхронно передают биты в *MS*, используя представление двоичных цифр 1 и 0 сигналами f_1 и f_0. Если *MS* при выполнении операции *И* получает только сигналы f_1, то результат наложения сигналов считается единицей. При приходе в *MS* сигналов f_1 и f_0 или f_0 результат равен нулю.

В операция *ИЛИ* при приходе в *MS* только сигнала f_0 результат равен нулю. При приходе в *MS* только сигнала f_1 или сигналов f_1 и f_0 результат равен единице.

Операции выполняются в *MS* без задержки сигнала за время, не зависящее от числа участников операции. Для вычислений *MS* не применяет логические элементы.

Операции *MAX* и *MIN*. Чтобы вычислить *MAX*, объекты передают в *MS* наибольшую цифру своего номера, которая представлена в виде шкалы из п. 4.1. В результате наложения шкал может появиться шкала с несколькими единицами в разных позициях шкалы. В ней объекты выбирают наибольшее значение цифры. Следующую цифру передают объектами, ранее передавшие наибольшую цифру, *MAX* определяется после передачи всех цифр чисел. Для расчета *MIN* объекты определяют минимальные значения цифр.

4.3. Распределенные аналого-цифровые операции

Рассмотрим операции аналого-цифрового суммирования. Эти операции существенно расширяют возможности цифровых операций (см. п. 4.2).

Добавим в модуль *MS* аналого-цифровой преобразователь (АЦП). Каждой цифре суммируемых чисел, представленной в произвольной системе счисления, выделим шкалу из п. 4.1. Шкалы цифр всех суммируемых чисел поступают в *MS* с побитным их совмещением. Приемник сигналов f_1 в *MS* для каждого бита шкалы объединяет энергию принятых сигналов f_1 и передает результат на АЦП, который выдает цифровое значение, соответствующее уровню принятой АЦП энергии, соответствующей энергии одного, двух и т. д. сигналов. Модуль *MS* отправляет эти числовые значения (частичные суммы) объектам. Объекты одновременно суммируют частичные суммы для получения окончательного результата.

Время для вычисления в *MS* состоит из двойного интервала времени передачи сигналов между *MS* и наиболее удаленным объектом и времени перевода «аналог — цифра».

Приведем пример сложения трех десятичных чисел $S = 68 + 58 + 68$. Они представлены шкалами: число 68 представлено шкалой младшего разряда R_1 = [010000000] и шкалой старшего разряда R_2 = [000100000]; число 58 представлено соответственно R_3 = [010000000] и R_4 = [000010000]; еще раз число 68 представлено R_5 = [010000000] и R_6 = = [000100000]. Вначале на *MS* поступают три шкалы R_2, R_4, R_6 младших цифр чисел, которые в результате наложения создают общую шкалу [0(3)0000000]. Здесь в круглых скобках указана суммарная энергия поступивших сигналов f_1. Эту шкалу АЦП переводит в цифровую шкалу ${}^1R$ = = [030000000], где цифра 3 указывает, сколько сигналов поступило в данный разряд от объектов *NC*. Эти данные направляются всем *NC*. Заметим, что энергия сигналов не обязательно идентична, и АЦП должен вносить соответствующую коррекцию. Для упрощения модуля связи АЦП может передавать в *NC* только цифровые измерения уровня энергии, где они будут переведены в число сигналов. Аналогично обрабатываются шкалы R_1, R_3, R_5 с образованием шкалы [000(2)10000], которая будет направлена объектам *NC* как ${}^2R$ = [000210000]. По шкалам ${}^1R$ и ${}^2R$ объекты получают из частичных сумм полную сумму $S = 10\ (2 \cdot 6 + 5) + 3 \cdot 8 = 194$.

Аналогично выполняется вычитание.

Часто полезно в АЦП только подсчитывать число полученных сигналов f_0. Это, например, позволит быстрее узнать, сколько объектов *NC* участвовало в операции.

Пример аналогового суммирования — гистограммы. Пусть группа *NC* оценивает некоторое событие по совокупности признаков. Каждому признаку объект *NC* присваивает количественное значение, и всю последовательность признаков *NC* передает в *MS* как единое сообщение — шкалу. Все шкалы передаются синхронно, с совмещением двоичных разрядов.

В процессе передачи в *MS* выполняется сложение. В результате все объекты получат гистограмму, каждый отсчет которой дает суммарную оценку конкретного параметра события. Число участни-

ков операции также легко подсчитывается, что даст гистограмму для средних значений параметров.

Часто признакам дается только двоичная оценка 1 и 0. Для получения гистограммы в этом случае сложение вырождается в операцию счета.

При участии n компьютеров в операциях из § 4 время выполнения каждой операции не зависит от числа участвующих в ней компьютеров и не отличается от времени с участием только одного или двух компьютеров (в зависимости от вида операции).

ЗАКЛЮЧЕНИЕ

Полученные результаты реализуют поставленные во Введении цели обеспечить быструю работу распределенного составного компьютерного кластера как распределенного компьютера. Кластер выполняет аппаратно с высокой скоростью следующие операции.

- Обеспечение точной синхронизации обменов сообщениями, позволяющей отправлять сообщения группы компьютеров получателям как единое общее сообщение двух видов: сообщение, состоящее из следующих одно за другим без временных задержек сообщений; сообщение, объединяющее одноименные двоичные разряды всех сообщений одинаковой разрядности в общее сообщение той же разрядности.
- Аппаратное выполнение барьерной синхронизация асинхронных процессов, обычно выполняемое программно.
- Устранение конфликтов доступа компьютеров к сети также аппаратно путем посылки короткого сообщения для одновременного разрешения группы конфликтов.
- Выполнение распределенных вычислений с одновременным участием в общей операции данных из многих сообщений. Выполнение операции во время передачи только одного сообщения с наложением одноименных двоичных разрядов.

Эти результаты решают поставленные во Введении задачи: сети должны эффективно работать с короткими сообщениями; сетевые средства должны быть простыми, не содержать компьютеры; при этом они должны выполнять распределенные вычислительные и управляющие операции в сетевых средствах с высокими скоростями, близкими к достижимым в компьютере.

Полученные решения могут быть полезны при создании алгоритмов обработки данных и качественно отличаются от известных. Обычно хороший алгоритм должен по возможности реже обращаться к сетевым средствам, применение которых весьма замедляет его выполнение. В рассмотренной структуре применение сети ускоряет выполнение алгоритма. Как показано в § 2 и 4 ускорению способствует замена поочередной передачи сообщений многих источников синхронизированной передачей единственного сообщения с совмещением одноименных разрядов всех сообщений.

Подчеркнем в завершение, что предложенные технические сетевые средства простые, не содержат программируемые устройства.

Дальнейшее направление исследований может быть связано с созданием применяющих сетевые обмены алгоритмов и оценкой их эффективности. Такая оценка представляет собой отдельную сложную задачу, требующая для ее решения участия специалистов, создающих сложные прикладные алгоритмы, и заинтересованных в их ускорении с привлечением новых технических средств.

ЛИТЕРАТУРА

1. *Tennenhouse, D.L.* Towards an Active Network Architecture // SIGCOMM Comput. Commun. Rev. — 1996. — 26 (2).
2. *Kreutz, D., Ramos, F.M.V., Verissimo, P.,* et. al. Software-Defined Networking. A Comprehensive Survey // Proc. of the IEEE. — 2015. — Vol. 103, no. 1, January. — P. 14—76.
3. *Tokusashi, Y., Huynh, Tu Dang, Pedone, F.,* et al. The Case for In-Network Computing on Demand // Dresden, Germany. EuroSys'19. March 25—28. — 2019. — https://doi.org/10.1145/3302424.3303979
4. *Sapio, A., Abdelaziz, I., Aldilaijan, A.,* et al. In-Network Computation is a Dumb Idea Whose Time Has Come // Proceedings of the 16th ACM Workshop on Hot Topics in Networks — HotNets-XVI. — 2017. — http://dx.doi.org/10.1145/3152434.3152461
5. *Ports, D.R.K., Nelson, J.* When Should the Network Be the Computer? // Workshop on Hot Topics in Operating Systems (HotOS'19), May 13—15, Bertinoro, Italy. — 2019. — https://doi.org/10.1145/3317550.3321439
6. *In-Network* Computing and Next Generation HDR 200G Infiniband Whitepaper // Mellanox Technologies. — 2018. — https://www.mellanox.com/pdf/whitepapers/WP_In-Network_Computing_Next_Generation_HDR_200G_IB.pdf
7. *Стецюра Г.Г.* Компьютерная сеть с быстрой распределенной перестройкой своей структуры и обработкой данных в процессе их передачи // Проблемы управления. — 2017. № 1. — С. 47—56. — http://pu.mtas.ru/archive/Stetsyura_117.pdf [*Stetsyura, G.G.* A Computer Network with Fast Distributed Reconfiguration and Data Processing During Transfer // Automation and Remote Control. — 2018. — Vol. 79, iss. 4. — P. 713—724. — DOI: 10.1134/S0005117918040124]
8. *Stetsyura, G.* Means for Fast Performance of the Distributed Associative Operations in Supercomputers // Communications in Computer and Information Science: Springer International Publishing AG. — 2017. — Vol. 793. — P. 27—39. — https://doi.org/10.1007/978-3-319-71255-0_3
9. *Стецюра Г.Г.* Сетевая информационно-вычислительная поддержка взаимодействия подвижных роботов // Проблемы управления. — 2018. — № 5. — С. 56—65. — DOI: http://doi.org/10.25728/pu.2018.5.6 [*Stetsyura, G.G.* Network Information-Computing Support of Automatic Mobile Objects Interaction // Automation and Remote Control. — 2019. —

Vol. 80, iss. 6. — P. 1134—1147. — DOI: https://doi.org/10.1134/S0005117919060110]

10. *Патент* РФ 2697729 C1. Способ и система управления взаимодействием автономных мобильных технических объектов с быстрой реакцией на изменение состояния объектов и внешней среды. [*Patent* RF. 2697729 C1. Method and control system for the interaction of autonomous mobile technical objects with a quick reaction to a change in the state of objects and the environment.]

Статья представлена к публикации членом редколлегии В.М. Вишневским.

Поступила в редакцию 03.12.2019, после доработки 15.04.2020. Принята к публикации 3.06.2020.

Стецюра Геннадий Георгиевич — д-р техн. наук,
Институт проблем управления им. В.А. Трапезникова РАН,
г. Москва, ✉ gstetsura@mail.ru.

THE COMPUTER CLUSTERS WITH FAST SYNCHRONIZATION OF MESSAGES AND WITH FAST DISTRIBUTED COMPUTING BY THE NETWORK HARDWARE

G.G. Stetsyura

V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences, Moscow, Russia
✉ gstetsura@mail.ru

Abstract. The network structure and methods for the rapid interaction of computers in a distributed composite cluster are proposed. The composite cluster is organized hierarchically and consists of a group of simple clusters, one of which gives tasks to simple clusters. Simple clusters perform tasks synchronously and asynchronously. In the simple cluster, a group of processors also acts synchronously or asynchronously, using fast barrier synchronization. The activities of simple cluster computers are controlled by a leading computer. The composite cluster quickly performs the processes of synchronization of messages sent, processes of resolving conflicts of computer access to network tools, distributed logical operations, the distributed definition of *max* and *min*, distributed addition and subtraction operations. These operations do not require message delay to complete. The duration of operations does not depend on the number of cluster computers participating in them. To do this, computers send messages simultaneously, creating a group message in which the bits of the same name are combined in time. Acceleration of the above mentioned distributed computing and synchronization is achieved with intensive computer access to the cluster network, which distinguishes the proposed solutions from the existing practice of using a computer network. The proposed operations allow creating faster algorithms for real-time tasks, including tasks for managing the cluster.

Keywords: computer cluster, hierarchical network structure, fast computing in the network, dynamic reconfiguration, distributed synchronization, barrier synchronization, the distributed accelerator of computing.