УДК 681.3

ОЦЕНКА ВРЕМЕНИ ОБРАБОТКИ ДАННЫХ В КЛАСТЕРНЫХ СИСТЕМАХ

В.Г. Тульчинский, А.К. Чарута

Институт кибернетики им. В.М. Глушкова НАН Украины
03680, Киев 187, проспект Академика Глушкова, 40
факс: 526 7418, т.: 526 2008, e-mail:aik@public.icyb.kiev.ua

Рассмотрено влияние распараллеливания на время решения задач, связанных с обработкой больших объемов данных (баз данных или файлов). Полученные оценки позволяют оптимизировать архитектуру программы для исполнения в кластерной системе на основе небольшого числа экспериментов.

The influence of parallel implementation of the data processing tasks on the task computation time is considered for databases and files. Obtained estimations allow optimize the program architecture for execution in a cluster system on the base of few experiment results.

Вступление

Современные параллельные вычислители с архитектурой МИМД (кластеры) предназначены для выполнения с максимальной эффективностью объемных вычислений. Важнейшей особенностью вычислений на кластере является максимальная концентрация вычислительных ресурсов всего кластера или его выделенной части для решения одной задачи. В результате параметры системы остаются стабильны в течение всего времени выполнения задачи. Приложение может быть заранее или в процессе работы оптимизировано с учетом значений этих устойчивых параметров. Такая архитектура делает кластеры привлекательным объектом для формального анализа и применения математических методов.

В работе рассматривается влияние способа распределения данных на производительность кластера в задачах, требующих интенсивной обработки больших объемов информации.

Цель работы

Значительную часть практических задач, решаемых на кластерах, составляют задачи однородной обработки больших массивов данных. Такие задачи, как правило, легко распараллеливаются, так как обладают естественной параллельностью по данным. Мы будем рассматривать именно распараллеленные задачи. Их особенность – синхронность полезных операций на всех узлах. Такие задачи возникают, в частности, в геологии / геофизике (обработка результатов сейсмических исследований земной коры), при обработке естественно-языковых текстов, изображений (например, результатов аэрофотосъемки и космоснимков земной поверхности), при обработке видео, в задачах Data Mining и статистической обработки больших баз данных.

Цель работы – оптимизация распределения данных и операций над данными с целью минимизации очередей и ускорения решения задачи всей системой.

Постановка задачи

Наиболее распространенная в наше время архитектура кластеров «Беовульф» [1], если отвлечься от деталей реализации, представляет собой соединенный высокопроизводительной сетью набор однотипных узлов – обычных ПК. Во многих случаях архитектура включает общее хранилище данных, например, дисковую стойку. Вариант архитектуры с общим диском может быть реализован без дисковой памяти на узлах.

Управляются такие кластеры, как правило, ОС Linux, для обмена сообщениями используются реализации MPI. (Можно запускать и программы, написанные под PVM, но PVM разработана для гетерогенных кластеров и в однородной среде работает медленнее.) При старте программы MPI создает несколько одинаковых процессов, как правило, по одному на каждый процессор каждого задействованного узла кластера. Процессы знают свой номер и общее число процессов задачи и в зависимости от номера выбирают алгоритм работы. Хотя MPI обеспечивает разнообразные способы передачи данных, включая широковещание и асинхронный обмен, реализованные на кластерах параллельные алгоритмы главным образом основаны на синхронном обмене. Процесс, посылающий данные, ожидает завершения их приема, а принимающий ожидает посылки и передачи данных. Операция приема данных имеет две модификации: прием от одного конкретного процесса или от любого процесса задачи.

Далее термин «процесс» используется в двух смыслах: как в информатике и как в математической статистике. Это удобно, так как речь идет об одном объекте, а смысл конкретного употребления термина ясен из контекста.

Рассмотрим классические стратегии распределения данных в кластерной системе [2]: централизация, дублирование, расчленение, смешанная стратегия.

Централизация означает, что единственная копия базы данных или файла управляется одним процессом на одном узле кластера или в дисковой стойке. С ростом числа узлов кластера и интенсивности обращений к данным централизованный доступ к данным становится узким местом системы.

Дублирование означает, что на всех или нескольких узлах расположены копии данных. На первый взгляд, индивидуальная работа процессов со своими локальными данными может полностью убрать очереди. Но это верно лишь для случаев полной независимости процессов по данным. Если же результаты, записанные одним процессом, могут быть затребованы другими процессами, каждое обновление данных необходимо распространить на все узлы. Кроме того, большие данные могут просто целиком не поместиться на одном узле.

Расчленение означает, что непересекающиеся подмножества данных управляются процессами на разных узлах. Эта стратегия хороша, если данные обладают иерархической или блочной структурой. В случае сложно структурированных данных ее реализация требует значительных усилий.

В наиболее популярной смешанной стратегии организуется несколько расчлененных копий данных. Еще одна смешанная стратегия – сегментация (централизация с дублированием части данных). Известно, что 80 % обращений к базе данных приходятся только на 20 % записей данных. При сегментации на узлах размещаются небольшие, но наиболее востребованные фрагменты данных.

При проектировании или настройке программы на параметры конкретной кластерной системы приходится также выбирать между специализацией нескольких узлов на обслуживании обращений к данным и сочетанием работы данными с вычислениями. Ясно, что оптимизация работы с данными в общем случае – довольно трудная задача. Для ее решения предлагается оценить время выполнения параллельного вычислителя заданного размера при решении задачи на основе небольшого числа экспериментально определяемых параметров.

Рассмотрим следующую модель параллельной обработки большого объема данных на кластере, на котором N процессов выполняются параллельно. Работа каждого из них может быть представлена в виде чередования расчетов и обращений к данным для чтения или записи. Еще S процессов (СУБД или файловые системы хранилища) обслуживают обращения к данным. Каждый из них ждет поступления запроса от одного из вычислительных процессов, затем выполняет его. Пока обслуживающий процесс выполняет запрос, на него могут поступить другие запросы. Все они ставятся в очередь неограниченной емкости и обслуживаются в порядке поступления. В результате вычислительный процесс n представляется последовательностью интервалов (неотрицательных действительных чисел) трех типов: *расчет* (t_i^n), *ожидание* освобождения данных (δ_i^n), *чтение или запись* данных (τ_i^n).

Поведение такой системы зависит от соотношения между временем расчетов и временем доступа к данным. Будем рассматривать два предельных случая: со слабой загрузкой, когда очередь запросов на доступ к данным одной операции успевает рассосаться до начала следующей операции доступа к данным, и с полной загрузкой, когда эта очередь, возникнув на первом шаге, не рассасывается до конца расчетов.

Условие слабой загрузки может быть сформулировано следующим образом: $\forall i \sum_n \tau_i^n << t_{i+1}^n$. В случае полной загрузки $\forall i \sum_n \tau_i^n \geq t_{i+1}^n$.

Для рассматриваемых задач различием времени выполнения на i-м шаге расчета или обращения к данным в зависимости от номера процесса можно пренебречь, поэтому опустим лишний индекс. Время, за которое будет закончено i шагов процесса n, обозначаем $\gamma_i^n = t_0 + \delta_1^n + \tau_1 + t_1 + \delta_2^n + \tau_2 + t_2 + ... + \delta_i^n + \tau_i + t_i$.

Условия слабой и полной загрузки можно, соответственно, переписать как $N\tau_i << t_{i+1}$ и $N\tau_i \geq t_{i+1}$.

Случай слабой загрузки

В этом случае увеличение числа параллельно работающих процессов с одной стороны уменьшает время расчетов t_i (в силу распараллеливания), а с другой – увеличивает число обращений к данным пропорционально N. В результате очередь быстро перестает рассасываться. Однако случай слабой загрузки наиболее интересен, так как после возникновения устойчивой очереди дальнейшее распараллеливание не имеет практического смысла.

Централизованный доступ со слабой загрузкой. **В случае централизованного доступа количество обслуживающих процессов $S = 1$. Обозначим также *накопленное* к i-му шагу *время ожидания процесса*** n $\Delta_i^n = \delta_1^n + \delta_2^n ... + \delta_i^n$. **Так как** $\delta_i^n \geq 0$

$$\Delta_{i+1}^n \geq \Delta_i^n . \tag{1}$$

Для удобства, не снижая общности, будем считать, что вычислительные процессы упорядочены по возрастанию δ_1: $\delta_1^{n+1} > \delta_1^n$. Тогда можем утверждать, что

$$\Delta_i^{n+1} \geq \tau_i n . \tag{2}$$

Это утверждение легко доказывается по индукции, так как из правила упорядочения следует $\gamma_1^{n+1} > \gamma_1^n$, а очередь запросов гарантирует, что раньше пришедший запрос будет обработан раньше, т.е. $\gamma_i^{n+1} \geq \gamma_i^n + \tau_i$, после чего достаточно просуммировать по n и учесть, что $\Delta_i^0 \geq 0$.

Из утверждений (1) и (2) получаем $\Delta_I^{n+1} \geq \max\limits_{1\leq i\leq I}\{\tau_i\}n$. Отметим, что небольшое изменение процедуры формирования очереди обеспечивает

$$\Delta_I^{n+1} = \max_{1\leq i\leq I}\{\tau_i\}n. \tag{3}$$

Для этого достаточно только соблюдать порядок шагов и не выполнять τ_{i+1} до того, как выполнены все τ_i. Средствами MPI это делается элементарно: вместо ожидания сообщения от любого процесса ставится команда ожидания сообщения от следующего процесса по модулю N.

Обозначим максимальное время выполнения обращения к памяти $\tau_{\max} = \max\{\tau_i\}$. В силу слабой загрузки можно полагать $N\tau_i < t_{i+1}$, т.е. очередь не накапливается. Поэтому общее время выполнения задачи γ_Σ вычисляется как время выполнения самого длинного потока: $T_N = \gamma_I^N = \sum t_i + \sum \tau_i + (N-1)\tau_{\max}$.

С практической точки зрения эту формулу удобно переписать по-другому, чтобы параметры можно было легко оценивать. Пусть при $N=1$ общие затраты на обработку данных $D = \sum \tau_i$, а расчеты выполняются за время $C = T_1 - D_1$. Обозначим долю чистых расчетов $\theta = C/T_1$, а долю распараллеливаемой части расчетов $0 < \eta \leq 1$. Тогда с учетом задержки (3) время работы программы на N процессов по формуле Амдала составляет:

$$T_N = (1-\eta)C + \frac{\eta C}{N} + D + (N-1)\tau_{\max}, \text{ откуда} \tag{4}$$

$$T_N = \left((1-\eta\theta) + \frac{\eta\theta}{N}\right)T_1 + (N-1)\tau_{\max}. \tag{5}$$

Параметр η определяется экспериментально по серии запусков одной задачи с разным числом процессов N.

Дублирование данных со слабой загрузкой. Простейший случай дублирования без размножения данных обобщает случай централизации. Пусть у нас есть S-процессорный сервер данных и на каждом процессоре выполняется процесс СУБД, обслуживающий как в централизованном случае до $\lceil N/S \rceil$ вычислительных процессов. Формула (5) преобразуется к виду

$$T_N = \left((1-\eta)\theta + \frac{\eta\theta}{N} + \frac{1-\theta}{S}\right)T_1 + \left(\lceil N/S \rceil - 1\right)\tau_{\max}. \tag{6}$$

Представляется, что размещение копий данных на всех узлах кластера может значительно уменьшить очереди. Запросы на чтение данных будут в этом случае выполняться без задержек. Однако запись промежуточных результатов потребует широковещательного оповещения всех узлов. В случае полного дублирования обработка должна быть синхронной, т.е. процессу на узле нужно кроме своих данных записать изменения на остальных $S-1$ обслуживающих процессах прежде, чем переходить к следующему шагу. Поэтому, обозначив затраты на чтение ρD и заменив D на $(\rho/S + (1-\rho))D$, формулу (4) можно преобразовать к виду

$$T_N = \left(1 - \rho + \theta(\rho - \eta) + \frac{\eta\theta}{N} + \frac{\rho(1-\theta)}{S}\right)T_1 + (S-1)\tau_{\max}^W, \tag{7}$$

где $\tau_{\max}^W$ – максимальное время записи данных, необходимых более чем одному процессу. В общем случае помимо T_N нужно учесть затраты на начальное копирование данных на все узлы.

Расчленение данных со слабой загрузкой. В этой стратегии непересекающиеся подмножества данных распределены по узлам кластера и управляются отдельными S процессами. Хотя пропорциональное расчленение (да и вообще эффективное расчленение) не всегда возможно, оно часто встречается в рассматриваемых задачах с естественной параллельностью. Пусть запросы выполняются в порядке поступления, а вероятность обращения конкретного запроса к k-му обслуживающему процессу $p_k = 1/S$. Рассмотрим процессы, описывающие входные запросы от N узлов кластера, как независимые. Хотя это не

совсем так, при достаточно больших S и N длина очереди в момент запроса, а, следовательно, и время ожидания будут случайным образом меняться. Предположим также, что все N процессов стационарны и ординарны. В силу синхронности операций каждого процесса вероятность скопления запросов от одного процесса равна нулю, т.е. все функции Пальма $\varphi_k(t) \equiv 1$. По той же причине интенсивность суммарного потока Λ примерно постоянна, и, значит, интенсивности каждого из N одинаковых потоков $\lambda_n = \lambda$ равномерно сходится к нулю с ростом N. Тогда имеем дело с суперпозицией потоков малой интенсивности. По предельной теореме теории массового обслуживания [3] на k-й обслуживающий процесс приходит простейший поток с интенсивностью $\Lambda = \lambda N$ (по теореме Пальма). Для этого случая известно общее решение А.Я. Хинчина из которого, в частности, среднее время ожидания

$$\mu(\delta) = \frac{1-\alpha}{2\alpha}\frac{\mu(\tau^2)}{\mu(\tau)}, \tag{8}$$

где α – вероятность простоя обслуживающего процесса, т.е. среднее время ожидания вызовов за единицу времени, μ – обозначение математического ожидания.

$$\mu(T_N) = (1-\eta)C + \frac{\eta C}{N} + \frac{D}{S} + \mu(I)\frac{1-\alpha}{2\alpha}\frac{\mu(\tau^2)}{\mu(\tau)}, \tag{9}$$

$$\mu((1-\alpha)\gamma_\Sigma) = D/S \Rightarrow \alpha = 1 - D/(S\mu(\gamma_\Sigma)). \tag{10}$$

С учетом $\mu(I) = D/(S\mu(\tau))$, (9) и (10) получаем квадратное уравнение, неотрицательный корень которого

$$\mu(T_N) = T_1\left(\Phi_N + \Delta_S + \sqrt{{\Phi_N}^2 + {\Delta_S}^2\frac{\mu(\tau^2)}{2\mu(\tau)^2}}\right), \tag{11}$$

где $\Phi_N = \frac{\theta}{2}\left(1-\eta+\frac{\eta}{N}\right)$, $\Delta_S = \frac{1-\theta}{S}$.

Формулу (11) легко модифицировать в зависимости от способа обработки данных для случаев $S = N$ и $S + N = const$.

При выводе формулы (11) применялись оценки для предельного случая, а не для заданного числа узлов, чтобы избежать решения системы уравнений и получить оценку в явном виде.

В отличие от случая дублирования в случае расчленения мы получили только среднюю оценку. Из теории массового обслуживания известно, что дисперсия времени ожидания отдельного вызова превышает его матожидание. Хотя общая дисперсия времени ожидания будет снижаться с увеличением времени работы (т.е. числа запросов к данным), стохастический характер этой величины нужно учитывать в тех случаях, когда программа запускается редко, или работает очень долго: максимальное время работы может быть важнее среднего времени.

Случай полной загрузки

При достаточно большом N задача переходит в состояние полной загрузки. В этом случае время работы задачи при расчленении данных

$$\mu(T_N) = \left(t_0^1 + t_i^N\right) + \frac{N}{S}D. \tag{12}$$

Для централизованного доступа

$$T_N = \left(t_0^1 + t_i^N\right) + \left\lceil\frac{N}{S}\right\rceil D. \tag{13}$$

При больших N $\mu\left(t_0^1 + t_i^N\right) \approx (t_0 + t_i)\left(1-\eta+\frac{\eta}{N}\right) \approx (t_0 + t_i)(1-\eta) = t_{const}$, причем для длинных задач $t_{const} << D$. Отсюда (12) и (13) сводятся в общую формулу

$$\mu(T_N) \approx \frac{N}{S}(1-\theta)T_1 + t_{const}. \tag{14}$$

Для дублирования данных оценка получается сложней, так как учитывает необходимость дублирования записи во все обслуживающие процессы:

$$T_N \approx \left\lceil \frac{N}{S} \right\rceil \rho D + N(1-\rho)D + t_{const}\,. \qquad (15)$$

Оценки (14), (15) показывают, что в зоне полной загрузки (при достаточно больших N) время работы программы начинает линейно расти с ростом N.

Численный эксперимент

В численном эксперименте использовалась имитация процесса «чистых» расчетов в сочетании с обращениями к данным для централизованного доступа (рис. 1, 2) и расчленения (рис. 3, 4). «Количество процессоров» на графиках обозначает N; «Эксперимент» – экспериментальные данные; «Слабая загрузка» – расчет по формуле (5) для рис. 1, 2 и по формуле (11) для рис. 3, 4; «Полная загрузка» – расчет по формуле (14) с $t_{const} = 0$.

Испытания централизованного доступа проводились на кластере СКИТ–2, в Институте кибернетики им. В.М. Глушкова НАН Украины, с общей дисковой памятью. Испытания расчленения данных проводились на кластере ИНПАРКОМ-16 на киевском заводе «Электронмаш», имеющем отдельные диски на узлах.

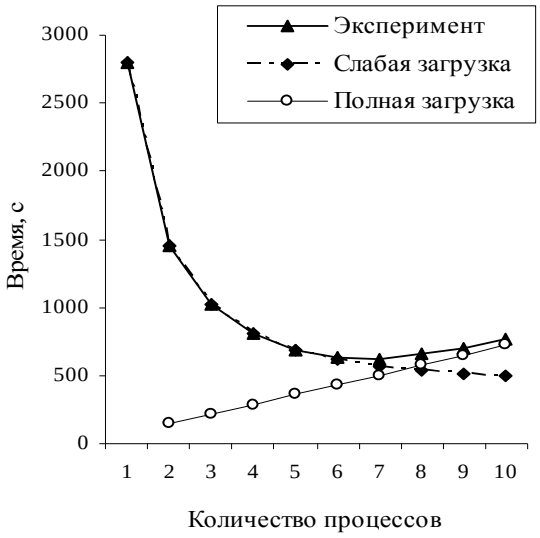

Рис. 1. $\theta = 0.975$, $\eta = 0.993$, $\tau_{\max} = 15$, $T_1 = 2794$

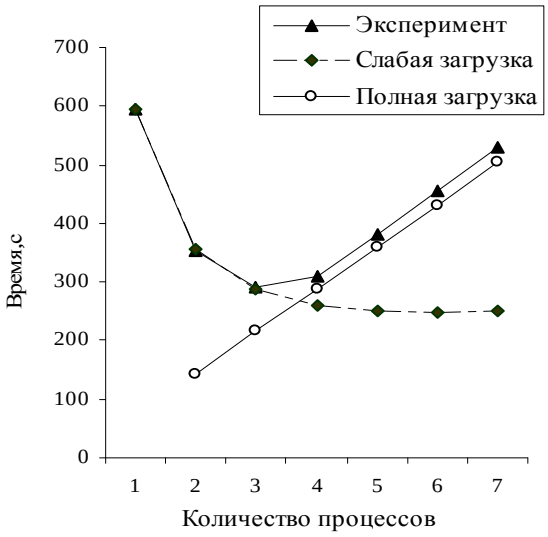

Рис. 2. $\theta = 0.884$, $\eta = 0.96$, $\tau_{\max} = 15$, $T_1 = 594$

Рис. 3. $S = 4$, $\theta = 0.966$, $\eta = 0.94$, $\mu(\tau) = 14$, $\mu(\tau^2) = 246$, $T_1 = 2821$

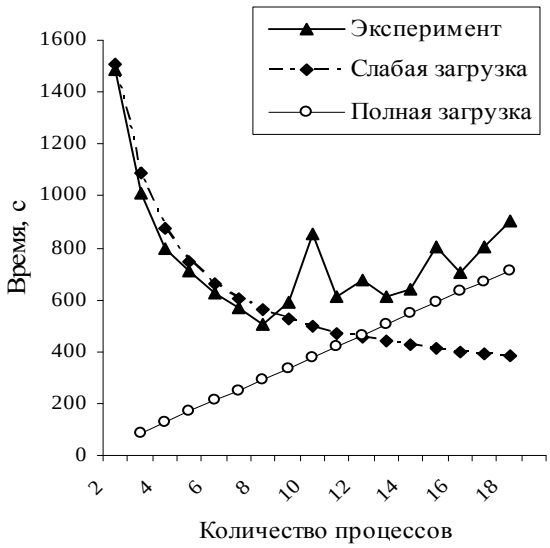

Рис. 4. $S = 2$, $\theta = 0.966$, $\eta = 0.93$, $\mu(\tau) = 14$, $\mu(\tau^2) = 246$, $T_1 = 2821$

Большие колебания экспериментального графика для расчленения данных связано со стохастическим характером зависимости в этом случае. Некоторое занижение оценок полной загрузки для больших N вызвано пренебрежением $t_0^1 + t_i^N$ и коммуникационными расходами.

Сравнение аналитических оценок с экспериментальными данными демонстрирует пригодность точки пересечения графиков слабой и полной загрузки в качестве хорошего начального приближения точки перегиба экспериментального графика производительности. Для ее уточнения достаточно провести экспериментальное исследование в небольшой окрестности точки пересечения. Эксперименты также продемонстрировали высокую точность оценок (6), (11) для слабой загрузки.

Выводы

Формулы (6), (7) и (11) в сочетании с формулой (14) позволяют по результатам небольшого числа простых экспериментов:

- оптимизировать распределение данных между узлами кластера;
- подбирать наилучшее количество обслуживающих и вычислительных процессов для достижения максимального быстродействия задачи.

За пределами рассмотрения остался смешанный случай и случай сегментации, возникающий как комбинация централизации и дублирования, если данные целиком не помещаются на узлах. Оба эти варианты могут быть оценены путем комбинирования и небольшой модификации полученных оценок. В работе дополнены и уточнены оценки, полученные в [4].

1. *Аладышев О.С., Савин Г.И., Телегин П.Н., Шабанов Б.М.* Кластеры класса Беовульф // Научно-технический журнал "Известия высших учебных заведений. Электроника". – 2001, № 1. – С. 7 – 13. (http://www.jscc.ru/informat/ClusterBeoWulf.html)
2. *Тиори Т., Фрай Дж.* Проектирование структур баз данных: В 2-х кн. Кн. 1. Пер. с англ. – М.: Мир, 1985. – 287 с.
3. *Хинчин А.Я.* Работы по математической теории массового обслуживания. – М.: Физматгиз, 1963. – 236 с.
4. *Тульчинский В.Г., Тульчинский П.Г.* Обработка больших объемов данных в кластерных системах // Искусственный интеллект. – 2005, № 4. – С. 651 – 657.