МАТЕМАТИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ МЕТОДА УПРЕЖДАЮЩЕГО КЭШИРОВАНИЯ

САХАРОВ И.Е., асп.

Предлагаются системная модель обращения к данным, критерий оценки эффективности применения упреждающего кэширования, организация имитационной модели модуля упреждающего кэширования и результаты тестирования разработанного метода.

In this article author suggests system model of data reading, performance assessment criterion of using prefetch caching. Also author submits structure of simulation model of prefetch caching module and some test results of this method.

Ключевые слова: кэширование, модель обращения к данным, оценка эффективности, имитационная модель.
Key words: caching, data reading model, performance evaluation, simulation model.

Системная модель обращения к данным для упреждающего кэширования. Метод упреждающего кэширования подробно описан в работе [1]. Чтобы оценить его эффективность, предлагается использовать имитационное моделирование процесса упреждающего кэширования. Эффективность вычислений будет измеряться временем выполнения приложения, которое обозначим T. В общем случае время T рассчитывается по формуле

$$T = N_{I/O}(\overline{T_{CPU}} + \overline{T_{I/O}}), \quad (1)$$

где $N_{I/O}$ – количество операций ввода-вывода; $\overline{T_{CPU}}$ – среднее время исполнения процессором приложения между двумя последовательными запросами к данным; $\overline{T_{I/O}}$ – среднее время на выполнение одного запроса на ввод/вывод.

Время выполнения запросов выборки данных зависит от того, где располагаются эти данные. Если данные располагаются в кэш-памяти, то пусть T_{HIT} – время выборки одного блока данных из кэш-памяти. Если блок данных не располагается в кэш-памяти, то он должен быть подкачен с дискового накопителя. Задержку выборки блока данных с диска обозначим T_{disk}, время передачи запроса по сети и время передачи блока данных – $T_{Network}$, а время обработки запроса удаленным компьютером или серверов – T_{server}. Для выборки буфера и обслуживания этого запроса требуется процессорное время T_{driver}. Таким образом, если блок данных располагается не в кэш-памяти, время выполнения запроса рассчитывается по формуле

$$T_{MISS} = T_{HIT} + T_{Network} + T_{server} + T_{disk} + T_{driver}. \quad (2)$$

Упреждающее кэширование старается максимально использовать время T_{MISS} для осуществления максимального количества предвыборок данных. Известно, что оперативная память является общей для какой-то вычислительной установки. Поэтому величина кэш-памяти, выделяющейся под упреждающую выборку, всегда будет ограничена. Операционные системы (ОС) стараются использовать оперативную память по максимуму для осуществления кэширования данных. Если под кэширование остается небольшая часть оперативной памяти, то возникают множественные ошибки памяти (page fault), что приводит к большим издержкам при выполнении приложения [2].

Пусть вся оперативная память представляет собой массив буферов одинакового размера. Для упреждающего кэширования выделяется n буферов. Также введем понятие коэффициента попадания в кэш $H(n)$, который будет функцией от n. Тогда среднее время на обработку одного запроса $\overline{T_{I/O}}$ вычисляется по следующей формуле:

$$\overline{T_{I/O}} = H(n)T_{HIT} + (1 - H(n))T_{MISS}. \quad (3)$$

На рис. 1 показана зависимость коэффициента попадания от объема кэш-памяти. Такой вид графика обусловлен тем, что большинство ОС используют алгоритм стека. Если взять первую производную по n от уравнения (3), то получим следующее:

$$\Delta\overline{T_{I/O}} \approx H'(n) \cdot (T_{HIT} - T_{MISS}). \quad (4)$$

Получается, что коэффициент попадания в кэш-память зависит от среднего времени доступа.

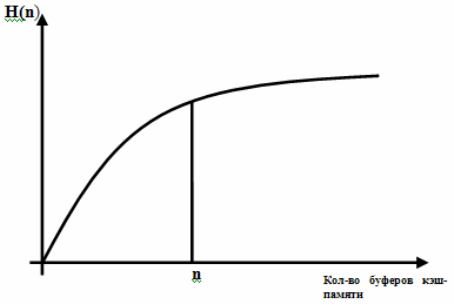

Рис. 1. График зависимости коэффициента попадания от объема кэш-памяти

В реальном процессе функция $H(n)$ будет представляться так, как показано на рис. 2. Обычно происходит резкий скачок коэффициента попадания после того, как большая часть требуемых данных располагается в кэш-памяти. Такой вид коэффициента попадания приводиться во многих источниках, в частности [3, 4].

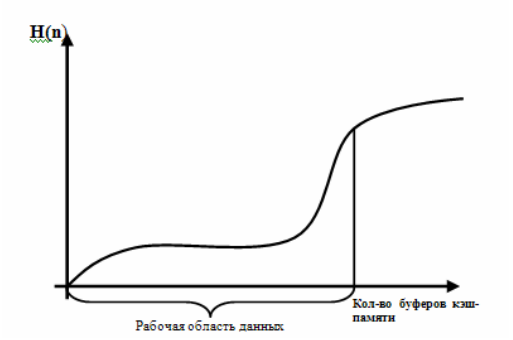

Рис. 2. График зависимости коэффициента попадания от объема кэш-памяти

Математическая оценка эффективности упреждающего кэширования. Очевидно, что выигрыш при упреждающей выборке достигается только для первого обращения к скаченному блоку, а повторное обращение к этому блоку не приведет к увеличению эффективности. Неважно был ли блок скачен заранее или скопирован по запросу. После первого обращения этот блок будет находиться в оперативной памяти (либо в кэш-памяти УК, либо в кэш-памяти операционной системы). Также надо заметить, что процессору приходиться выполнять дополнительные дисковые операции и операции поиска (T_{driver}).

Можно ограничить эффективность предвыборки данных сверху. Пусть осуществляются предвыборки некоторого количества блоков, что соответствует некоторому количеству x запросов данных. Тогда максимальное время, за которое подкачивается блок данных, равно $T_{Network} + T_{disk} + T_{server}$. Так как процесс выполнения приложения является последовательным процессом, то сумма $T_{Network} + T_{disk} + T_{server}$ является максимальным выигрышем от применения упреждающего кэширования. Следовательно, модуль упреждающего кэширования формирует последовательность будущих обращений к файлам один раз, и эта последовательность является неполной (по отношению ко всем запросам к данным, встречающимся в приложении) и ограниченной некоторым числом запросов на предвыборку. По мере выполнения приложения генерируется обращение к файлам. Часть требуемых блоков уже находится в кэш-памяти, но чем больше происходит обращений к файлам, тем меньший эффект имеет заранее исполненная последовательность обращений к данным. Поэтому требуемые данные перестают находиться в кэш-памяти, что влечет запуск процессов копирования этих данных. Поэтому максимальный выигрыш от применения упреждающего кэширования для одной сформированной последовательности обращений к файлам не будет превышать $T_{Network} + T_{disk} + T_{server}$.

Допустим, что приложение осуществляет x запросов к данным. Время выполнения каждого запроса складывается из $\overline{T_{CPU}}$ – среднего времени исполнения процессором приложения между двумя последовательными запросами к данным, T_{HIT} – минимального времени доступа к блоку данных и T_{driver}. То есть время выполнения x запросов вычисляется по формуле

$$x(\overline{T_{CPU}} + T_{HIT} + T_{driver}). \quad (5)$$

Обозначим через B_x временное преимущество от применения упреждающего кэширования, тогда

$$B_x = (T_{disk} + T_{network} + T_{server}) - \\ -x(\overline{T_{CPU}} + T_{HIT} + T_{driver}). \quad (6)$$

Предположим, что $\overline{T_{CPU}}$ и T_{driver} стремятся к нулю, тогда формула (6) принимает следующий вид:

$$B_x = (T_{disk} + T_{network} + T_{server}) - x \cdot T_{HIT}. \quad (7)$$

Рассмотрим вариант, при котором B_x = 0, тогда формула (7) приобретает вид

$$P = x = (T_{disk} + T_{network} + T_{server}) / T_{HIT}, \quad (8)$$

где P – горизонт предвыборки (определяет количество запросов, после которых предвыборка становится не эффективной).

Итак, на основе полученных оценок производится анализ эффективности упреждающего кэширования, а горизонт предвыборки используется для генерации последовательности предвыборок и правильного расчета размера очереди упреждающих вызовов (рис. 3).

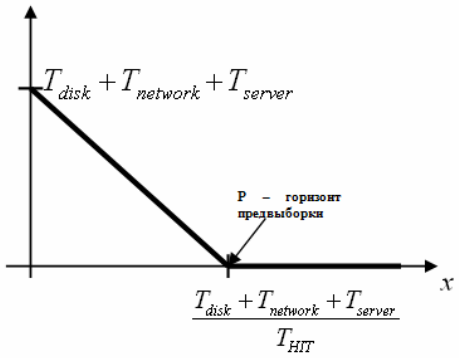

Рис. 3. Функция выигрыша от применения упреждающего кэширования

Имитационная модель. На основе системной модели обращения к данным была создана имитационная модель модуля упреждающего кэширования на языке GPSS [4]. Модуль упреждающего кэширования представляет собой многоканальную систему массового обслуживания с ожиданием [1]. Модель поделена на сегменты. Изначальным транзактом является так называемый транзакт-паспорт задания. От него создаются копии, а сам он уничтожается. Число копий равно количеству узлов, на которых запускается задача. Каждый из созданных транзактов-паспортов занимает свой узел и остается в нем до момента полного выполнения задания на данном узле.

На каждом узле от каждого транзакта создаются копии, число которых равняется числу запрашиваемых файлов. Эти новые транзакты определяют процессы подкачки начальных блоков необходимых файлов. После процесса подкачки начальных блоков каждый транзакт займет место в соответствующих кэш-памятях, т.е. теперь транзакты моделируют подкаченные порции блоков.

После завершения подкачки начальных блоков на каждом узле создаются транзакты-строки, которые моделируют строки программы. Каждая транзакт-строка обращается к определенным блокам, в случае отсутствия в кэш-памяти необходимых блоков происходит однократное копирование транзакта-строки с заданием необходимых параметров. Этот новый транзакт имитирует вызов процесса подкачки необходимых блоков. Когда все транзакты-строки для узла будут обработаны в модели (программа на конкретном узле закончит свое выполнение), происходит освобождение узла транзактом-паспортом. Когда все начальные транзакты-паспорты освободят свои узлы, моделирование заканчивается.

Модель состоит из 12 сегментов:

1. Задание начальных параметров функционирования модели.
2. Создание паспорта задания.
3. Таблицы метаданных.
4. Формирование паспорта для вычисляющих узлов.
5. Подкачка и выгрузка блоков в кэш-память.
6. Локальная сеть кластера.
7. Внешняя сеть связи с другими узлами.
8. Формирование строк программы.
9. Выполнение программы на узлах.
10. Запуск процессов подкачки блоков.
11. Завершение работы вычислительных узлов.
12. Вспомогательный сегмент.

Имитационное моделирование показало, что применение механизма упреждающего кэширования увеличивает эффективность распределенной обработки при определенных параметрах функционирования модели. Максимальный прирост производительности был получен при следующих параметрах функционирования механизма: на выполнение 300 операций ввода-вывода было потрачено ~25000 единиц модельного времени (см. рис. 4), что составляет ~60% от времени выполнения задания без применения механизма УК.

Такой выигрыш от применения упреждающего кэширования достигается при периодически постоянных вызовах к данным.

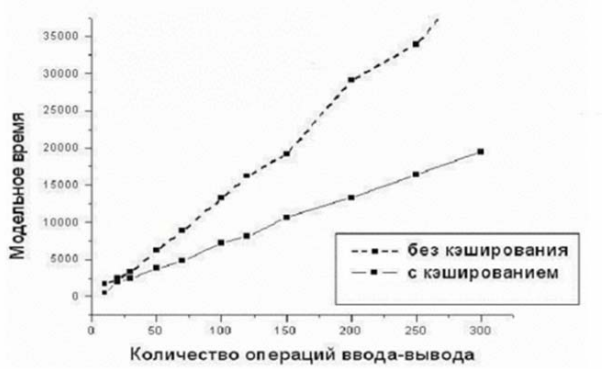

Рис. 4. Результаты имитационного моделирования упреждающего кэширования

Оценка производительности модуля упреждающего кэширования. Для оценки эффективности упреждающего кэширования использовались как стандартные тестовые пакеты, с внесением в них небольших изменений, так и собственные разработанные приложения. Набор тестовых приложений отражает большой спектр приложений с интенсивным вводом/выводом и дает полную картину возможностей применения упреждающего кэширования.

Все используемые данные для тестовых приложений располагаются на множестве узлов ввода/вывода. Доступ к данным обеспечивает файловая система. В работе использовалась файловая система PVFS. Запуск производился на тестовом вычислительном кластере под операционной системой Linux с ядром версии 2.4. Основной параметр, по которому оценивается эффективность упреждающего кэширования, – это время исполнения приложения. Также использовалось множество других показателей, дополняющих друг друга и предоставляющих дополнительную информацию о процессе исполнения приложений. В тестовом наборе присутствуют однопотоковые и многопотоковые приложения. С помощью экспериментов доказана эффективность упреждающего кэширования для широкого класса приложений. Одной из особенностей разработанного модуля упреждающего кэширования является возможность работы с многопотоковыми приложениями. Эксперименты выявили достоинства и недостатки упреждающего кэширования. Главным недостатком метода упреждающего кэширования на основе разделения потоков является его зависимость от реализации анализатора приложений. Автоматический процесс формирования упреждающего потока может содержать не все инструкции, относящиеся к вводу/выводу. Чем больше пропущенных инструкций, тем менее эффективно упреждающее кэширование. Существенное влияние на эффективность метода оказывает наличие сложных зависимостей по данным в приложении, что влечет за собой множественную синхронизацию. Необходимо заметить, что при последовательных обращениях к данным упреждающее кэши-

рование лишь замедляет процесс исполнения приложения. Нет никакого смысла применять упреждающее кэширование для приложений, состоящих в основном из последовательных обращений к файлам. Первоначальная идея применения упреждающего кэширования состоит в использовании метода упреждающего кэширования там, где файловая система не может обеспечить требуемой эффективности выполнения приложения. Реальный прирост производительности наблюдается при сложных последовательностях обращений к файлам (непоследовательное чтение, зависимое чтение от состояния некоторых переменных, обращение к данным с учетом некоторой формулы, одновременное чтение множества файлов).

Тестирование. Для проверки разработанного модуля упреждающего кэширования использовались следующие тестовые приложения: XDataSlice – средство визуализации трехмерных массивов; FFT – модифицированное приложение для расчета преобразования Фурье; Agrep – программа поиска в текстовых файлах, тестовые приложения для СУБД Postgres; Mpich2 – перемножение двух матриц, файлы которых располагаются на множестве узлов ввода/вывода (см. таблицу).

Тестовое приложение	Без исп. УК, сек	С исп. УК, сек*	Кол-во дисковых операций, участвовавших в предвыборке, %	Эффективность УК,%	Процессорное время, сек	Синхр.,%	Упрежд. поток,%
XDataSlice Вариант 1	687,95	312,76 (32,59)	62,14	**54,53**	240,47	0,06	0,18
XDataSlice Вариант 2	831,05	645,95 (37,22)	12,99	**22,27**	151,18	0,76	0,09
XDataSlice Вариант 3	364,87	317,23 (30,02)	66,61	**13,06**	253,93	0,02	0,19
XDataSlice Вариант 4	302,99	296,78 (32,02)	30,00	**2,04**	150,46	2,04	0,20
FFT 256 Кб	331,42	335,74 (0,00)	0,00	**-1,30**	244,70	2,16	0,12
FFT 512 Кб	662,68	673,84 (0,00)	0,00	**-1,68**	546,75	1,35	0,06
Agrep	23,4	17,28 (3,01)	65,02	**26,15**	20,84	0,0	0,02
Postgres 20	86,53	65,44 (8,47)	34,53	**24,37**	31,58	1,10	0,05
Postgres 80	231,42	201,12 (45,18)	47,32	**13,09**	95,15	2,25	0,12
Mpich2	534,45	415,34 (39,71)	74,30	**22,28**	130,48	7,96	0,74

Сахаров Илья Евгеньевич,
Академия ФСБ России,
научный сотрудник,
тел. 8-917-547-18-38, e-mail: siebox@mail.ru.

Ilya Sakharov,
research officer of Federal Security Service Academy,
telephone 8-917-547-18-38, e-mail: siebox@mail.ru.

Модуль упреждающего кэширования поддерживает специальную программную предвыборку данных. Результаты экспериментов показали, что для класса приложений, для которых упреждающее кэширование потенциально эффективно, прирост производительности варьируется в пределах от 15 до 60 %.

Список литературы

1. Сахаров И.Е. Метод упреждающего кэширования на основе разделения потоков в высокопроизводительных распределенных вычислительных системах. – Кострома: Изд-во КГТУ, 2008.

2. Patterson R.H. Informed Prefetching and Caching CMU-CS-97-204, - December 1997.

3. A trace-driven comparison of algorithms for parallel prefetching and caching / T. Kimbrel, A. Tomkins, R.H. Patterson // Proceedings of the Second USENIX Symposium on Operating Systems Design and Implementation, USENIX. – October 1996. – P. 19–34.

4. Томашевский В.Н., Жданова Е.Г. Имитационной моделирование в среде GPSS. – М.: Бестселлер, 2003. – С. 5–24.