Федор Викторович ЛАКТИОНОВ —
зав. сектором образовательных информационных систем
mega.t72@gmail.com

Олег Валерьевич АНДРЕЕВ —
зав. кафедрой неорганической и физической химии, доктор химических наук, профессор
andreev@utmn.ru

Вадим Анатольевич ФИЛИППОВ —
проректор по новым образовательным и информационным технологиям, кандидат социологических наук
filippov-vadim@yandex.ru

Тюменский государственный университет

УДК 519.172.1

*ПРИМЕНЕНИЕ АВЛ-ДЕРЕВЬЕВ В УПРАВЛЕНИИ ДИНАМИЧЕСКОЙ ПАМЯТЬЮ**

APPLICATION OF AVL-TREES IN THE MANAGEMENT OF DYNAMIC MEMORY

АННОТАЦИЯ. Опираясь на алгоритмы построения сбалансированных деревьев, с учетом их основной функции — динамической балансировки узлов, предлагается использовать их вместо линейных ассоциативных списков в задаче распределения динамической памяти, что может значительно повысить производительность некоторых классов вычислительных задач.

SUMMARY. Based on the algorithms for balanced trees, with the consideration of their main function – dynamic balancing nodes, the authors of the article encourage to use them instead of linear associative lists in the problem of dynamic memory allocation, which can significantly improve the performance of certain classes of computational tasks.

КЛЮЧЕВЫЕ СЛОВА. Динамическая память, АВЛ-деревья, куча.
KEY WORDS. Dynamic memory, AVL-trees, heap.

Динамическая память на сегодняшний день — неотъемлемая часть любого программного продукта, и, поскольку характер использования такой памяти непредсказуем, на диспетчер управления динамической памятью накладываются особые требования, главным из которых является скорость выделения и освобождения ресурса.

В Windows для управления динамической памятью существутт специальный инструмент — диспетчер управления кучей (heap manager) или диспетчер динамической памяти. Куча (heap) — это область зарезервированного адресного пространства размером в одну или более страниц, из которой диспетчер кучи может динамически выделять память меньшими порциями [1, 2]. С помощью диспетчера динамической памяти любому приложению становится доступно использовать практически не ограниченный по размерам и числу набор виртуальных ресурсов [1] и, следовательно, очень широкий спектр задач, решаемых на прикладном уровне.

* Работа выполнена при поддержке ФЦП «Научные и научно-исследовательские кадры инновационной России» на 2009-2013 гг. ГК П646.

Цель работы: проанализировать основные действия диспетчера динамической памяти и предложить алгоритм выделения и освобождения, основанный на АВЛ-деревьях, провести сравнительный анализ предложенных методов.

Анализ проблемы. Опираясь на теорию страничной организации виртуальной памяти [2, 3, 4], можно выявить следующий порядок действий, требуемый при выделении очередного ресурса динамической памяти:

1. Найти среди пула зарезервированных ресурсов кучи такой блок, размер которого удовлетворяет исходным требованиям.
2. Если такого блока не найдено, выделить дополнительный минимальный набор виртуальных страниц, который удовлетворяет запросу.
3. Зафиксировать для найденного блока объем используемой и неиспользуемой памяти.
4. Если из неиспользуемой памяти можно сформировать новый зарезервированный ресурс, то разделить блок по этому критерию, а вновь созданный вернуть в пул зарезервированных ресурсов.
5. Если новый блок сформировать нельзя, то пометить этот блок для дальнейшей оптимизации.
6. Сохранить и упорядочить найденный блок по признаку неиспользуемого объема памяти «сверху» и «снизу» блока (для возможности дальнейшей фрагментации-дефрагментации кучи).
7. Увеличить счетчик байт всех виртуальных регионов, которые пересекает выделенный блок.

При освобождении ресурса последовательность действий диспетчера может быть следующей:

1. Освободить блок от каких-либо оптимизаций.
2. Вернуть блок в пул зарезервированных ресурсов.
3. Уменьшить счетчик байт всех виртуальных регионов, пересекающий данный блок памяти.
4. Если счетчик стал равен нулю, что выявляет неиспользуемый регион — освободить весь диапазон страниц, его составляющих; фрагментировать свободный блок, покрывающий освобождаемый диапазон.

Большинство операций, проводимых диспетчером динамической памяти, заключается в поиске блока памяти. Большое число таких блоков приводит к тому, что проход по линейным спискам, которые лежат в основе стандартного диспетчера динамической памяти [1, 2], начинает составлять основную часть работы диспетчера. Кроме того, фрагментация виртуального пространства, связанная с неоднородностью запросов к диспетчеру, приводит к перерасходу памяти.

Предложенный алгоритм и его применение. Если вместо линейных списков использовать АВЛ-деревья поиска [5], можно рассчитывать не только на логарифмический подъем производительности диспетчера, но и на минимизацию фрагментации виртуальной памяти.

Структура блока в таком случае может быть следующей:

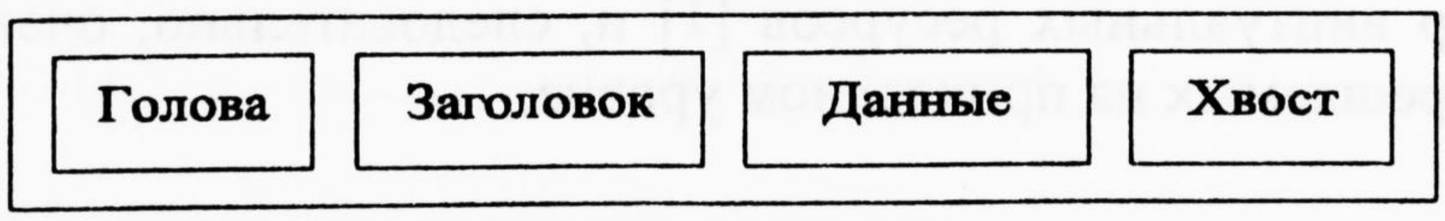

Рис. 1. Структура блока

«Голова» и «хвост» — это неиспользуемые фрагменты блока, их размер не может превышать размер заголовка, поскольку в этом случае из них можно

выделить в пул неиспользуемых ресурсов полноценные блоки, которые можно использовать в дальнейшем. «Заголовок» — это служебная информация блока, к которой относится также размер блока. Эта информация требуется как при освобождении блока, так и в процессе дефрагментации кучи. «Данные» — фрагмент памяти, ради которого и был произведен запрос диспетчеру.

«Хвост» — наиболее обычная ситуация, когда в пуле ресурсов отсутствует блок, точно соответствующий запрашиваемому размеру, но есть блок с небольшим перевесом — хвостом.

«Голова» — это ситуация, возникающая в момент, когда требуется освободить регион страниц, вслед за которым идет некоторый объем свободной памяти, но на организацию полноценного блока ее не хватает, поэтому этот объем добавляется к ближайшему распределенному блоку перед его заголовком — «в голову»

Чтобы осуществлять быстрый поиск свободного блока, можно использовать либо хэш из некоторого числа линейных списков, соответствующих размеру блоков [2, 6], которые в них хранятся, либо организовать дерево, критерием поиска которого станет размер блока. Поэтому заголовок свободного блока должен хранить информацию либо для списка, либо для АВЛ-узла дерева: левую и правую ссылки соответствующих поддеревьев и баланс узла.

Дефрагментация кучи — это объединение свободных блоков, расположенных непосредственно друг за другом. Чтобы найти такие последовательные блоки, можно использовать линейные списки, тогда поиск не будет занимать лишнее время, но в таком случае появится необходимость проходить линейный список в поисках позиции для каждого нового блока, встраиваемого в пул ресурсов диспетчера.

Для реализации списков дефрагментации блоков можно также использовать АВЛ-деревья. В этом случае критерием поиска такого дерева станет адрес заголовка блока.

Дефрагментация свободных блоков не является полноценной, поскольку любой блок «без хвоста», предшествующий блоку «с головой», всегда можно расширить, поглотив неиспользуемый фрагмент, и аналогично — любой блок без головного фрагмента можно растянуть на предшествующий «хвост», реализуя таким образом максимально доступное виртуальное пространство.

Дефрагментация «головных» и «хвостовых» фрагментов может быть осуществлена, если при встраивании очередного блока в пул ресурсов кучи при отсутствии последовательных блоков «слева» и/или «справа», будет производиться поиск соответствующего левого и правого блока «с хвостами». В этом случае для реализации полноценной дефрагментации понадобятся еще два дерева.

Поскольку «хвост» и «голова» актуальны только при фиксированном адресе данных блока, который соответствует распределенному ресурсу, то в этом блоке не нужно хранить информацию ни об узлах этих деревьев, ни о самих фрагментах.

Таким образом, заголовки зарезервированного и выделенного блока в пуле ресурсов должны иметь следующую архитектуру (табл. 1).

В один и тот же момент для любого блока требуется поддержка только двух деревьев. Несмотря на то, что размер блока на основе деревьев превышает размер блока со списками почти в три раза (20/8; 22/8), постепенная оптимизация виртуального пространства на АВЛ-узлах компенсирует этот недостаток.

Таблица 1

Архитектура заголовков блоков

Поле	Критерий	Поле	Критерий
Адрес левого блока	Адрес «заголовка»	Адрес левого блока	Адрес «головы»
Адрес правого блока		Адрес правого блока	
Баланс		Баланс	
Адрес левого блока	Размер блока	Адрес левого блока	Адрес «хвоста»
Адрес правого блока		Адрес правого блока	
Баланс		Баланс	
Размер блока		Размер блока	
		Размер «головы»	
		Размер «хвоста»	

В ходе работы над диспетчером было произведено несколько видов экспериментов:

• первый — выделение максимального числа блоков **фиксированного размера**. Этот эксперимент характеризует спектр задач, элементная база которых состоит из однотипных объектов, таких как списки и деревья. В ходе эксперимента внимание уделялось только скорости выделения ресурса. На рис. 2 и 3 заметна разница на хронологической шкале, которая отражает результат работы одного и того же алгоритма распределения ресурсов, но с разными диспетчерами динамической памяти.

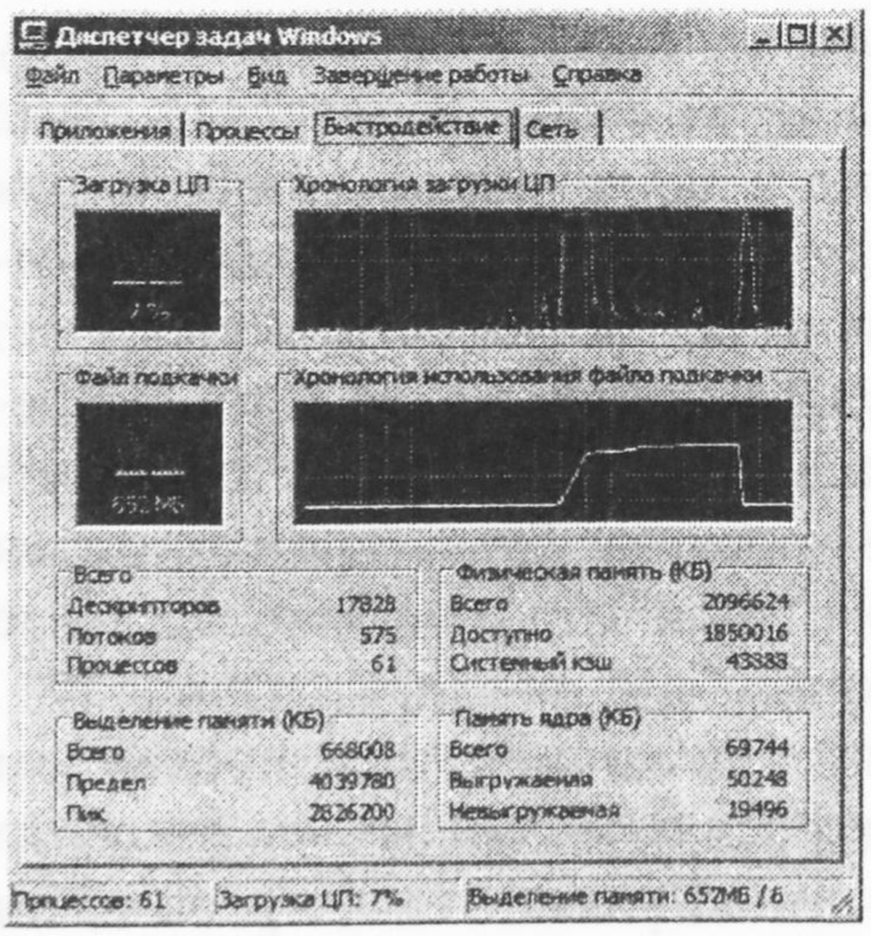

Рис. 2. АВЛ-диспетчер

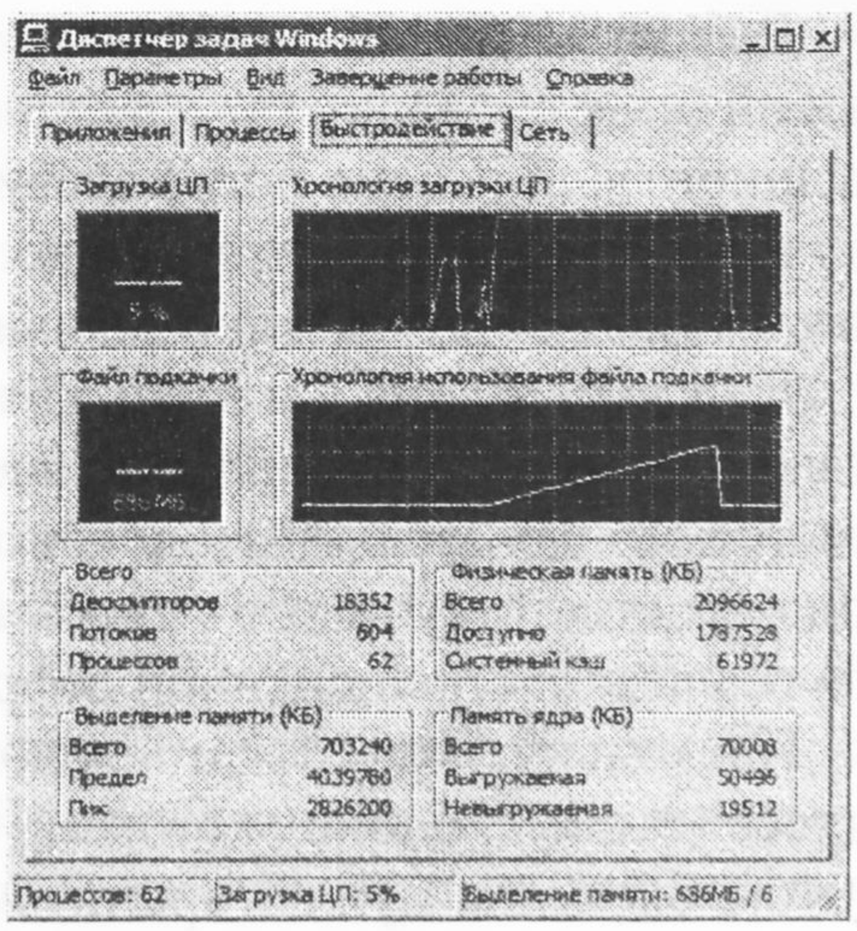

Рис. 3. Стандартный диспетчер

• следующий эксперимент — последовательное выделение **и освобождение** максимального числа блоков случайного размера: от 1 до 120 байт. В данном случае стандартный диспетчер имеет преимущество, поскольку его хэш на 128 списков не требует поиска свободного блока, в то время как АВЛ-алгоритм его производит. Здесь выигрыш АВЛ обуславливается не скоростью распределения ресурса, а скоростью его освобождения, где вступает

в силу логарифмический поиск позиции блока в дереве. В табл. 2 показаны результаты этого эксперимента.

• последний эксперимент — **случайное** выделение и освобождение максимального числа блоков случайного размера: от 1 байта до 120. Это самый сложный тест, который выявляет наиболее универсальное решение. Стандартный диспетчер в данном случае оказывается всегда быстрее, опережая АВЛ в 2-3 раза. Число освобождений в данном случае равно числу выделений, следовательно, падение скорости АВЛ может быть связано с неэффективностью критериев сортировки узлов, которые не учитывают интенсивность идентичных запросов.

Таблица 2

Результаты второго эксперимента

Номер подхода	Операция	АВЛ-диспетчер (сек.)	Стандартный диспетчер (сек.)
1	Распределение	17,625	16,406
	Освобождение	12,171	155,328
2	Распределение	20,484	15,00
	Освобождение	13,282	87,766

Выводы

При использовании АВЛ-деревьев можно рассчитывать на повышение производительности стандартного диспетчера динамической памяти в 5-6 раз, что вызвано уменьшением количества операций при поиске свободного блока. Понижение производительности до 3 раз происходит из-за неэффективности критериев сортировки узлов АВЛ-деревьев. Эффективность применения АВЛ-деревьев в управлении динамической памятью может быть достигнута при решении задач, динамика разнородных операций кучи в которых носит монотонный характер.

СПИСОК ЛИТЕРАТУРЫ

1. Managing Heap Memory in Win32: http://msdn.microsoft.com/en-us/library/ms810603.aspx.

2. Анализ встроенных механизмов защиты от переполнения кучи в Windows XP SP2: ttp://www.securitylab.ru/analytics/216376.php.

3. Dynamic memory allocation: http://en.wikipedia.org/wiki/Dynamic_memory_allocation.

4. Virtual memory: http://en.wikipedia.org/wiki/Virtual_memory.

5. Лактионов Ф.В., Филиппов В.А., Андреев О.В. Журнал научных публикаций аспирантов и докторантов. 2008. № 4. С. 207-210.

6. Кормен Т., Лайзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. М.: ИД Вильямс, 2005. 1296 с.