CORE
Metadata, citation and similar papers at core.ac.uk

統計解析環境Rの機能拡張について

2010年7月9日　統計数理研究所　オープンハウス

中野　純司　　データ科学研究系　教授

1　はじめに

統計ソフトウェアR (http://www.r-project.org/)は種々の計算機上で稼働するフリーソフトウェアであり, 統計学者の基本的なツールと言える. さらに最近になって, 非常に多くの分野のデータ解析にも利用されるようになっている.

計算機の進歩につれて, Rはかなり大きなデータも処理できるようになっている. ただし, Rの設計では使い易さを優先して, 実行時に命令を解釈するインタプリタ方式を採用していることもあり, その処理速度は決して早いとは言えない.

そのため, Rの高速化が望まれている. 現在, 並列計算を用いて高速化を計るのが一般的であるが, 本稿では、メモリー管理の改良を行う.

2　Rのメモリー管理

Rのメモリー管理は特定の計算機に依存しない汎用的なプログラムで書かれており, 例えば, ガベージコレクション上のページサイズは2Kbyteの固定となっている. 2Kbyteを超える物はmalloc関数によって, 別途ページを割り当てている. そしてそれぞれのページや記憶域内の境界は8byte固定で境界合わせを行っている. 比較的利用者の多いOSではページは2Kbyte単位で管理されているが, ページが4Kbyteや8KbyteのOSでは不利であることは明らかである. そして, ページングに関しては, 基本的には malloc 経由で確保しているので効率的に管理しているとは言いにくい. 速度を重視するなら, 理想的にはガベージコレクションも含め, brk, mmap等システムに最適化されたシステムコールを用いて, メモリー管理を行うのがよい. ただしこれには大幅なプログラミングを必要とする. そこで, 近年のCPUやOSがもつ特性を効果的に利用する.

2.1　大規模メモリー対策

最近の多くのOSには, 巨大メモリーを高速に扱う為の機能として Translation look aside buffer (TLB) などと呼ばれる機能がある. これは, システムによって異なった名称が用いられている. 例えば、SolarisではISM(Intimate Shared Memory), LinuxではHuge TLB, AIX と WindowsではLarge Pageと呼ばれる. ただ, Windowsの場合は, 起動直後でなければ連続領域を確保できないので, RDBMSなどサーバーアプリケーション向け機能と言って良い.

通常, 仮想メモリーをもつOSではアプリケーションが扱う論理アドレス空間を2Kbyte, 4Kbyte, 8Kbyte の単位でページとして扱い, 物理アドレスに変換するページテーブルをもつ. Memory management unit (MMU) はこのページテーブルのキャッシュとしてTLBをもっている. ページテーブルのキャッシュミスが発生した場合, それをTLB-missと呼び, OSとMMUがページテーブルを参照してアドレス変換を行い, 正しいメモリーにアクセスする.

今日ではパーソナルコンピュータでも2Gbyte程度は普通にメモリーを備えており, ページテーブルのエントリ数は非常に多くなっている. もし, 通常より大きな単位でページを利用すると, エントリ数を減らすことができ, TLBの有効利用もできる. ISMのような機能は, 論理アドレス空間を2〜256Mbyte(CPUにより異なる) 単位で扱う事ができ, 大規模メモリーを効率的に管理できる.

われわれはISM関数として, 大規模なページを扱う機能の実装を, R本体に対する試験的なパッチとして開発した. 大規模データを扱う場合, この関数を使えば, 管理するページエントリ数を削減し, TLB-missを減少させることができる. なお, 小規模なデータに対してこの関数を利用すると, 必要以上に大きいページを作成することになり, 時間的にも容量的にも無駄が発生することに注意しなくてはならない.

2.2　メモリとキャッシュの境界合わせ

現在のRは歴史的な理由により8byte単位で境界合わせを行っている.

今日のCPUの高速な命令, 例えばStreaming SIMD Extensions(SSE)等, には16byte境界でなければ動作しないものがある. また最適化BLASのひとつであるGotoBLAS2 (http://www.tacc.utexas.edu/tacc-projects/)は16byte境界のみサポートしていたが, R のために 8byte 境界もサポートしてくれた（しかし, 本来の性能が若干犠牲になっていると考えられる）. もちろん, 今現在のRで境界を 16byte に合わせただけでは, これらの最適化の恩恵を直ぐに受けられるわけではない.

Rで演算を行うとき, 多くのデータはメモリ上に置かれている. しかし, 高速に演算できるのはコアとレジスタの間である. レジスタとメモリの間には2層ないし3層のキャッシュが存在し, より上位のキャッシュは高速だが, 下位のキャッシュは低速である. また, 下位のキャッシュはコア間で共有しているケースもある. CPUとメモリ間における処理の単位を見ると, 例えばメモリーからノースブリッジ間にはデュアルチャンネルなら128bit単位でI/Oが発生する. ノースブリッジ-キャッシュ間は例えば256bit単位でアクセスが行われる.したがって, これらの単位に近い16byteのほうが, 8byteより境界合わせの単位として望ましい.

Rはオブジェクトをメモリにもつが, それはヘッダー部とベクトル部のフラットな構成になっており, ヘッダー部の情報を元にRインタプリタが各種操作を行う. 外部ライブラリを呼ぶ場合は, ベクトル部のポインターを渡して処理させている. 一般的にはベクトル処理は時間がかかるので, ベクトル部をできるだけキャッシュに乗せる方がキャッシュの効率が上がる. このためにも16byte単位の境界合わせがよい.

以上の考察より, 領域の確保部分およびヘッダー部分のオフセットを16byte境界にすることによって, もっとも遅いメモリーとの境界のずれによる遅延の軽減を試みた.すなわち, メモリー機構全体の改修までは行わなかったが, 境界サイズを標準の8byteから16byteに変更するパッチを作成し, 演算機構に対する効果を検証した.

3　ベンチマーク

広く知られているR-benchmark (http://r.research.att.com/benchmarks/) を用いて検証を行った. 外部ライブラリとしては, 性能が安定している並列化 GotoBLAS2 を用いた.

3.1　処理時間

時間による計測には, 若干のばらつきがあるので, それらの標準偏差も合わせて出せるようにR-benchmarkを改変した. 試行回数は16回とした.

表１, 表２を見ると, われわれのメモリ管理機構の改良によって全体で10%以上の高速化が計られたことになる.

3.2　プロファイリングによるCPUのイベントの採取

LinuxにはOprofileと呼ばれるCPUのプロファイリングを行うソフトウェアがある. これを用いて, TLBやCACHEのイベントを採取した.

表３, 表４に示されているとおり, 境界合わせ, ISM関数の利用とも, 望ましくないイベント数の減少につながることが確認された.

なお, 現在の境界合わせのパッチは単純なオブジェクトの境界を合わせたに過ぎないので, 他の部分での非効率性が高い可能性があることを注意しておく.

表１　R-benchmark25 Linux amd64 (AMD Opteron 2427 @ 2.2GHz)

	Original		Align16			ISM			ISM + Align16		
I. Matrix calculation	mean	sd	mean	sd	speed up	mean	sd	speed up	mean	sd	speed up
Creation, transp., deformation of a 2500x2500 matrix	1.55	0.18	1.52	0.36	1.7%	1.22	0.19	27.1%	1.19	0.36	30.3%
2400x2400 normal distributed random matrix ^1000	0.94	0.01	0.92	0.00	1.7%	0.89	0.01	5.7%	0.88	0.01	6.8%
Sorting of 7,000,000 random values	1.20	0.01	1.24	0.01	-3.5%	1.20	0.01	-0.1%	1.23	0.01	-2.7%
2800x2800 cross-product matrix (b = a' * a)	0.38	0.00	0.38	0.00	0.5%	0.31	0.00	23.7%	0.32	0.03	20.8%
Linear regr. over a 3000x3000 matrix (c = a \ b')	0.34	0.04	0.22	0.01	54.1%	0.36	0.04	-5.8%	0.23	0.01	48.9%
Trimmed geom. mean (2 extremes eliminated):	0.76		0.76		-0.5%	0.73		3.8%	0.69		9.3%
II. Matrix functions											
FFT over 2,400,000 random values	1.07	0.01	1.06	0.01	1.1%	0.90	0.00	19.0%	1.09	0.01	-1.8%
Eigenvalues of a 640x640 random matrix	1.91	0.06	1.89	0.07	0.7%	1.90	0.07	0.2%	1.88	0.07	1.3%
Determinant of a 2500x2500 random matrix	0.28	0.00	0.27	0.00	3.5%	0.26	0.00	10.9%	0.28	0.00	0.9%
Cholesky decomposition of a 3000x3000 matrix	0.28	0.00	0.28	0.01	0.5%	0.26	0.00	7.7%	0.28	0.00	1.7%
Inverse of a 1600x1600 random matrix	0.24	0.01	0.23	0.00	6.9%	0.25	0.00	-1.3%	0.22	0.00	9.5%
Trimmed geom. mean (2 extremes eliminated):	0.44		0.43		1.7%	0.39		12.4%	0.44		0.3%
III. Programmation											
3,500,000 Fibonacci numbers calculation (vector calc)	1.51	0.11	1.43	0.02	5.1%	1.53	0.09	-1.2%	1.40	0.01	8.0%
Creation of a 3000x3000 Hilbert matrix (matrix calc)	1.34	0.06	1.18	0.00	13.5%	1.11	0.06	20.6%	1.18	0.00	13.9%
Grand common divisors of 400,000 pairs (recursion)	5.12	0.33	3.84	0.12	33.2%	5.59	0.32	-8.5%	3.84	0.12	33.4%
Creation of a 500x500 Toeplitz matrix (loops)	1.07	0.04	1.15	0.05	-6.9%	1.01	0.04	6.4%	1.14	0.05	-5.5%
Escoufier's method on a 45x45 matrix (mixed)	0.99	0.05	0.96	0.00	2.9%	0.99	0.05	0.2%	0.96	0.00	3.8%
Trimmed geom. mean (2 extremes eliminated):	1.30		1.25		3.6%	1.20		8.2%	1.23		5.1%
Total time for all 15 tests	18.22		16.60		9.8%	17.77		2.6%	16.10		13.2%
Overall mean (sum of I, II and III trimmed means/3)	0.76		0.74		1.6%	0.70		8.1%	0.72		4.8%

表２　R-benchmark25 Linux amd64 (Intel Xeon CPU E5430 @ 2.66GHz)

	Original		Align16			ISM			ISM + Align16		
I. Matrix calculation	mean	sd	mean	sd	speed up	mean	sd	speed up	mean	sd	speed up
Creation, transp., deformation of a 2500x2500 matrix	0.75	0.16	0.71	0.28	6.3%	0.62	0.16	21.7%	0.60	0.28	25.1%
2400x2400 normal distributed random matrix ^1000	0.67	0.00	0.67	0.00	-0.1%	0.65	0.00	2.1%	0.66	0.00	0.9%
Sorting of 7,000,000 random values	0.93	0.01	0.92	0.01	1.2%	0.92	0.00	1.0%	0.91	0.00	1.7%
2800x2800 cross-product matrix (b = a' * a)	0.40	0.00	0.40	0.00	0.2%	0.38	0.00	5.9%	0.38	0.01	5.4%
Linear regr. over a 3000x3000 matrix (c = a \ b')	0.37	0.03	0.27	0.00	35.6%	0.36	0.04	3.1%	0.27	0.00	35.3%
Trimmed geom. mean (2 extremes eliminated):	0.58		0.57		2.1%	0.53		9.6%	0.53		9.9%
II. Matrix functions											
FFT over 2,400,000 random values	0.86	0.01	0.86	0.01	-0.6%	0.78	0.00	10.1%	0.87	0.01	-1.4%
Eigenvalues of a 640x640 random matrix	2.22	0.09	2.11	0.09	5.1%	2.10	0.08	5.3%	2.11	0.08	4.8%
Determinant of a 2500x2500 random matrix	0.30	0.00	0.30	0.00	0.7%	0.28	0.00	7.4%	0.30	0.00	0.6%
Cholesky decomposition of a 3000x3000 matrix	0.30	0.00	0.29	0.00	1.3%	0.27	0.00	11.2%	0.29	0.00	1.3%
Inverse of a 1600x1600 random matrix	0.28	0.00	0.28	0.00	0.0%	0.27	0.00	1.4%	0.28	0.00	-0.1%
Trimmed geom. mean (2 extremes eliminated):	0.42		0.42		0.5%	0.39		8.9%	0.42		0.1%
III. Programmation											
3,500,000 Fibonacci numbers calculation (vector calc)	1.11	0.08	1.08	0.00	3.4%	1.11	0.08	0.3%	1.08	0.00	2.8%
Creation of a 3000x3000 Hilbert matrix (matrix calc)	0.58	0.06	0.44	0.00	33.2%	0.50	0.06	17.7%	0.44	0.00	33.0%
Grand common divisors of 400,000 pairs (recursion)	3.05	0.30	1.86	0.06	63.9%	3.23	0.29	-5.5%	1.86	0.06	63.9%
Creation of a 500x500 Toeplitz matrix (loops)	0.84	0.04	0.84	0.04	1.0%	0.81	0.04	4.0%	0.85	0.04	-0.7%
Escoufier's method on a 45x45 matrix (mixed)	0.72	0.04	0.68	0.00	5.4%	0.73	0.04	-1.2%	0.68	0.00	5.8%
Trimmed geom. mean (2 extremes eliminated):	0.88		0.85		3.2%	0.87		1.0%	0.86		2.6%
Total time for all 15 tests	13.37		11.69		14.3%	13.00		2.9%	11.59		15.4%
Overall mean (sum of I, II and III trimmed means/3)	0.60		0.59		1.9%	0.57		6.4%	0.58		4.1%

表３　R-benchmark25 Linux amd64 Oprofile events (AMD Opteron 2427 @ 2.2GHz)

CPU	EVENT	Original	Align16	ISM	ISM + Align16
AMD Opteron 2427	CYCLES _NO _FPU _OPS _RETIRED	49163	37864	47735	35816
	DISPATCH _STALL _FOR _RESERVATION _STATION _FULL	18486	13731	19182	13817
	DISPATCH _STALL _FOR _SEGMENT _LOAD	35	35		1
	DISPATCH _STALL _WAITING _FOR _ALL _QUIET	28			1
	DISPATCH _STALLS	37042	31747	35825	28795
	INSTRUCTION _CACHE _MISSES	35	32	25	36
	L1 _DTLB _AND _L2 _DTLB _MISS	81	73	16	
	L1 _DTLB _MISS _AND _L2 _DTLB _HIT	65	68	52	47
	L2 _CACHE _MISS	279	247	214	211
	MAB _REQUESTS	19	10	12	15
	MAB _WAIT _CYCLES	105	84	85	97
	PAGE _TABLE _WALKER _1	3	2		
	PAGE _TABLE _WALKER _2	159	156	39	25
	RETIRED _INDIRECT _BRANCHES _MISPREDICTED	17	11	14	13
	RETURN _STACK _OVERFLOWS	4	3	5	5

表４　R-benchmark25 Linux amd64 Oprofile events (Intel Xeon CPU E5430 @ 2.66GHz)

CPU	EVENT	Original	Align16	ISM	ISM + Align16
Intel Xeon E5430	BR _TKN _BUBBLE _1	925	216	170	180
	BR _TKN _BUBBLE _2	434	687	401	661
	BUS _TRAN _RFO	22	18	24	17
	DTLB _MISSES	459	405	283	243
	IFU _MEM _STALL	1262	858	658	799
	L1D _M _REPL	131	108	178	167
	L1D _PEND _MISS	57748	46271	57584	45726
	L1I _MISSES	34	19	22	30
	L2 _IFETCH	68	47	46	60
	L2 _ST	82	71	122	108
	LLC _MISSES	153	123	152	125
	MEMORY _DISAMBIGUATION	3	2	3	2
	MISALIGN _MEM _REF		11		
	PAGE _WALKS	1678	1427	722	517

4　おわりに

本稿ではLinux上での動作を詳しく解析したが, 他のOS上でもほぼ同様の効果が得られている.

本稿で述べたようなソフトウェア内部の考察は, 普通のRのユーザにはほとんど興味のないものであろう. しかし, Rが遅いという不満はよく耳にする. Rはフリーソフトウェアであるので, 誰かが実際にプログラミングしない限り, 改善されることはない. われわれが行った試みは, 速度の劇的な改善をもたらすものではないし, 使い方によっては改悪にもなりうるものである. しかし, このような試みを積み重ねることによってのみ, フリーソフトウェアとしてのRがより使いやすくなる. 今後も, Rにとって汎用的で良いメモリー管理機構はどのようなものであるかを継続して調査・研究していきたいと考えている.

謝辞

本研究は株式会社COM-ONEの中間栄治氏との共同研究である. また, テキサス大学在職中の後藤和茂氏にはRとGotoBLASのデバッグを通して多くのアドバイスを頂いたことを深く感謝する. 本研究は, 情報・システム研究機構新領域融合研究センタープロジェクト「機能と帰納：情報化時代にめざす科学推論の形」によって実施された.