Data Formats for Dense Linear Algebra Algorithms speeds up the computation and reduce the data storage with a historical introduction

Fred Gustavson[1] and Jerzy Waśniewski[2](speaker)

1) IBM Research Center, Yorktown Height

e-mail: fg2@us.ibm.com

2) Technical University of Denmark

e-mail: jw@imm.dtu.dk

Slides:

http://www.imm.dtu.dk/~jw/Lectures/pmaa06.pdf
http://www.imm.dtu.dk/~jw/Lectures/pmaa06.ps.gz

August 29, 2006

Outline of my talk

- **A historical introduction**
- **Our previous development in this area**
- **The new algorithm RFP**
  "Rectangular Full Packed format"

A Look Back of DLA Software

- Machine language subroutines
- Very useful: Computer Journal and Communication of ACM
- Manufacture and homemade libraries
- The first good formulation: Handbook for automatic computation by J. Wilkinson and C. Reinsch
- Harwell, ESSL, NAG, ISML, CERN libraries
- There were many other libraries too
- Predecessors: Linpack, Eispack and level 1 BLAS
- Standard: levels 1, 2 and 3 BLAS
- Standard: LAPACK and ScaLAPACK

LAPACK and ScaLAPACK books

- **E. Anderson, Z. Bai, C. Bischof, L.S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. "LAPACK Users' Guide", Third edition. Society for Industrial and Applied Mathematics, 1999, Philadelphia, PA.**
- **L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley. "ScaLAPACK Users' Guide". Society for Industrial and Applied Mathematics, 1997, Philadelphia, PA.**

DLA books

- J. Dongarra, I. Duff, D. Sorensen, and H. van der Vorst. "Numerical Linear Algebra for High-Performance Computers", Software, Environments, Tools. Society for Industrial and Applied Mathematics, 1997, Philadelphia, PA.
- G. Golub and C. Van Loan. "Matrix Computations", Third edition. The Johns Hopkins University Press, Baltimore and London, 1996.
- There are several other good books. For example: Cleve Moler, Higham, Demmel, or Trefethen.

Present Standards of DLA

- LAPACK with Level 1, 2, and 3 BLAS (ATLAS, Goto)
  - Usually for single processors
  - Shared memory versions
  - Open MP
  - Partly for distributed memory versions
- ScaLAPACK with PBLAS, BLACS and MPI
  - Distributed memory computers
  - Network computing
- Harwell, NAG, IMSL, NetLib and others
- The whole system of MATLAB
- Computer vendor Libraries, i.e. ESSL and SunPerf

Symmetric and triangular matrices

- **Full storage**

n $= 7,$ **lda** $= 9,$ **memory needed** $=$ **lda** $\times$ **n** $= 63$

$$\begin{pmatrix}
a_{1,1_1} & \diamond & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{2,1_2} & a_{2,2_{11}} & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{3,1_3} & a_{3,2_{12}} & a_{3,3_{21}} & \diamond & \diamond & \diamond & \diamond \\
a_{4,1_4} & a_{4,2_{13}} & a_{4,3_{22}} & a_{4,4_{31}} & \diamond & \diamond & \diamond \\
a_{5,1_5} & a_{5,2_{14}} & a_{5,3_{23}} & a_{5,4_{32}} & a_{5,5_{41}} & \diamond & \diamond \\
a_{6,1_6} & a_{6,2_{15}} & a_{6,3_{24}} & a_{6,4_{33}} & a_{6,5_{42}} & a_{6,6_{51}} & \diamond \\
a_{7,1_7} & a_{7,2_{16}} & a_{7,3_{25}} & a_{7,4_{34}} & a_{7,5_{43}} & a_{7,6_{52}} & a_{7,7_{61}} \\
\oslash & \oslash & \oslash & \oslash & \oslash & \oslash & \oslash \\
\oslash & \oslash & \oslash & \oslash & \oslash & \oslash & \oslash
\end{pmatrix}$$

Symmetric and triangular matrices

- **Packed storage**

  $\mathbf{n} = 7, \quad \mathbf{memory\ needed} = \mathbf{n} \times (\mathbf{n}+1)/2 = 28$

$$\begin{pmatrix}
a_{1,1_1} & & & & & & \\
a_{2,1_2} & a_{2,2_8} & & & & & \\
a_{3,1_3} & a_{3,2_9} & a_{3,3_{14}} & & & & \\
a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\
a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\
a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\
a_{7,1_7} & a_{7,2_{13}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}}
\end{pmatrix}$$

Example; packed data storage

$$A = \begin{pmatrix} \underline{a_{11}} & a_{12} & a_{13} & a_{14} \\ a_{21} & \underline{a_{22}} & a_{23} & a_{24} \\ a_{31} & a_{32} & \underline{a_{33}} & a_{34} \\ a_{41} & a_{42} & a_{43} & \underline{a_{44}} \end{pmatrix}$$

$a_{ij} = \mathsf{conjg}(a_{ji})$ for $i, j = 1, \cdots, 4.$

UPLO = 'U'

a_{11}	a_{12}	a_{22}	a_{13}	a_{23}	a_{33}	a_{14}	a_{24}	a_{34}	a_{44}

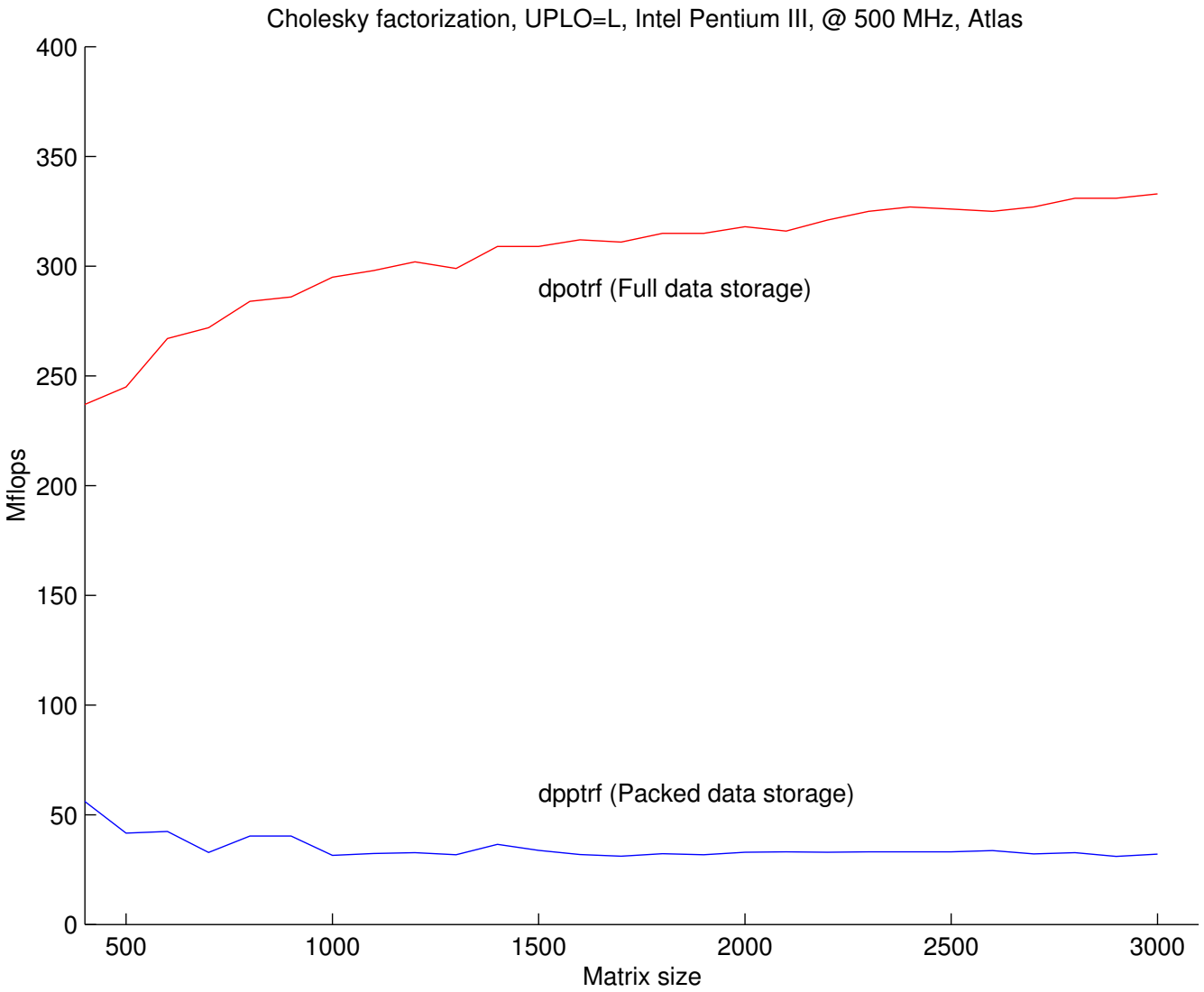
UPLO = 'L'

a_{11}	a_{21}	a_{31}	a_{41}	a_{22}	a_{32}	a_{42}	a_{33}	a_{43}	a_{44}

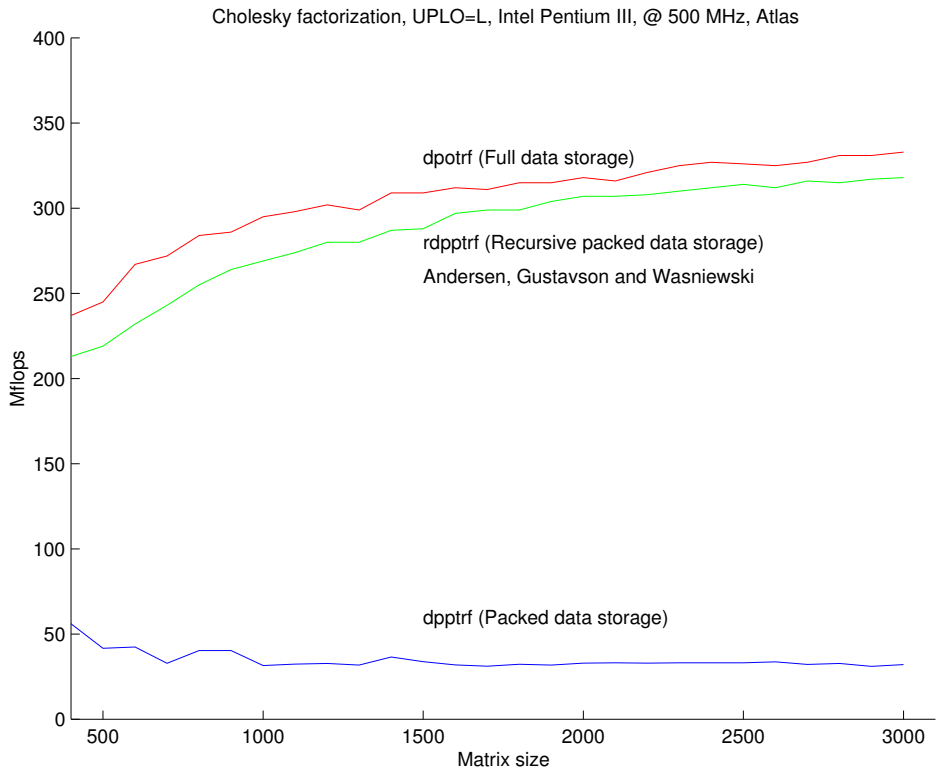
$size(\mathbf{AP}) = n(n+1)/2 = 10$

If $n = 1000$ **then** $size(\mathbf{AP}) = 500500$**, saving** 499500

Cholesky, a packed and full data storage

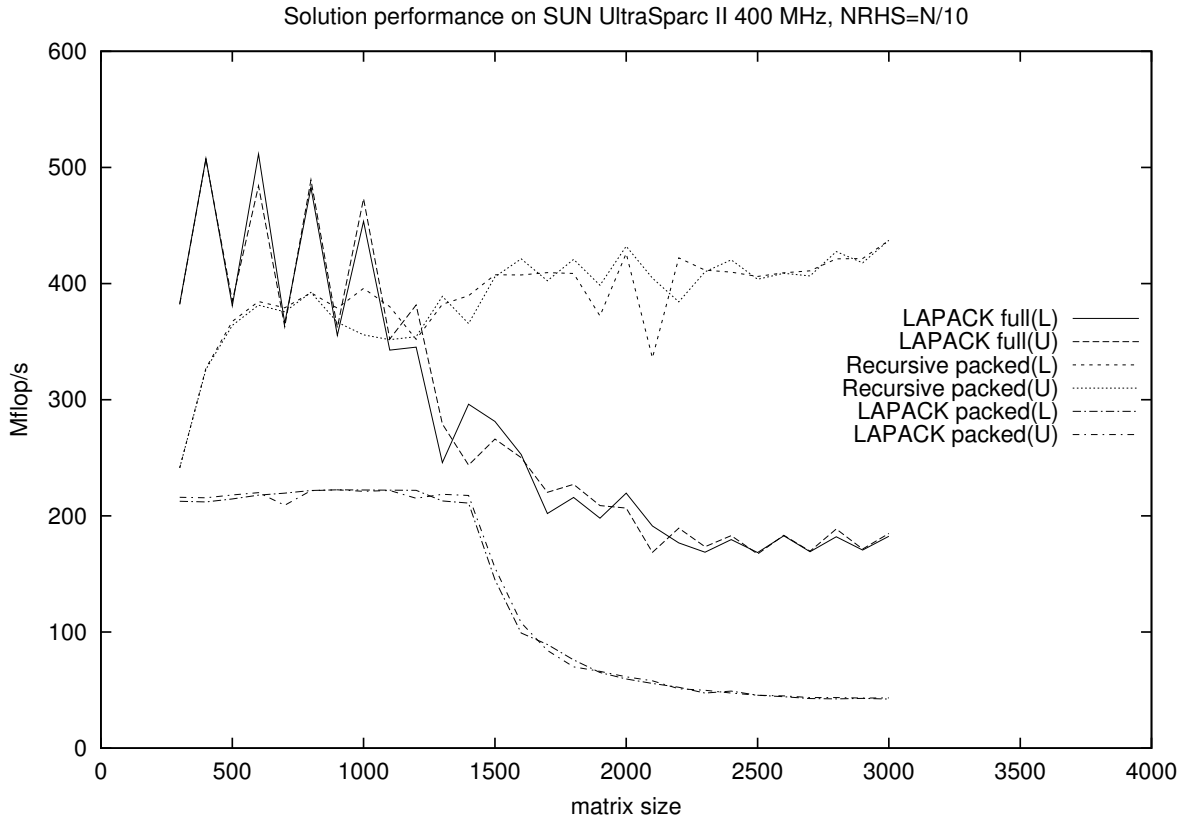


Cholesky, a packed and full data storage



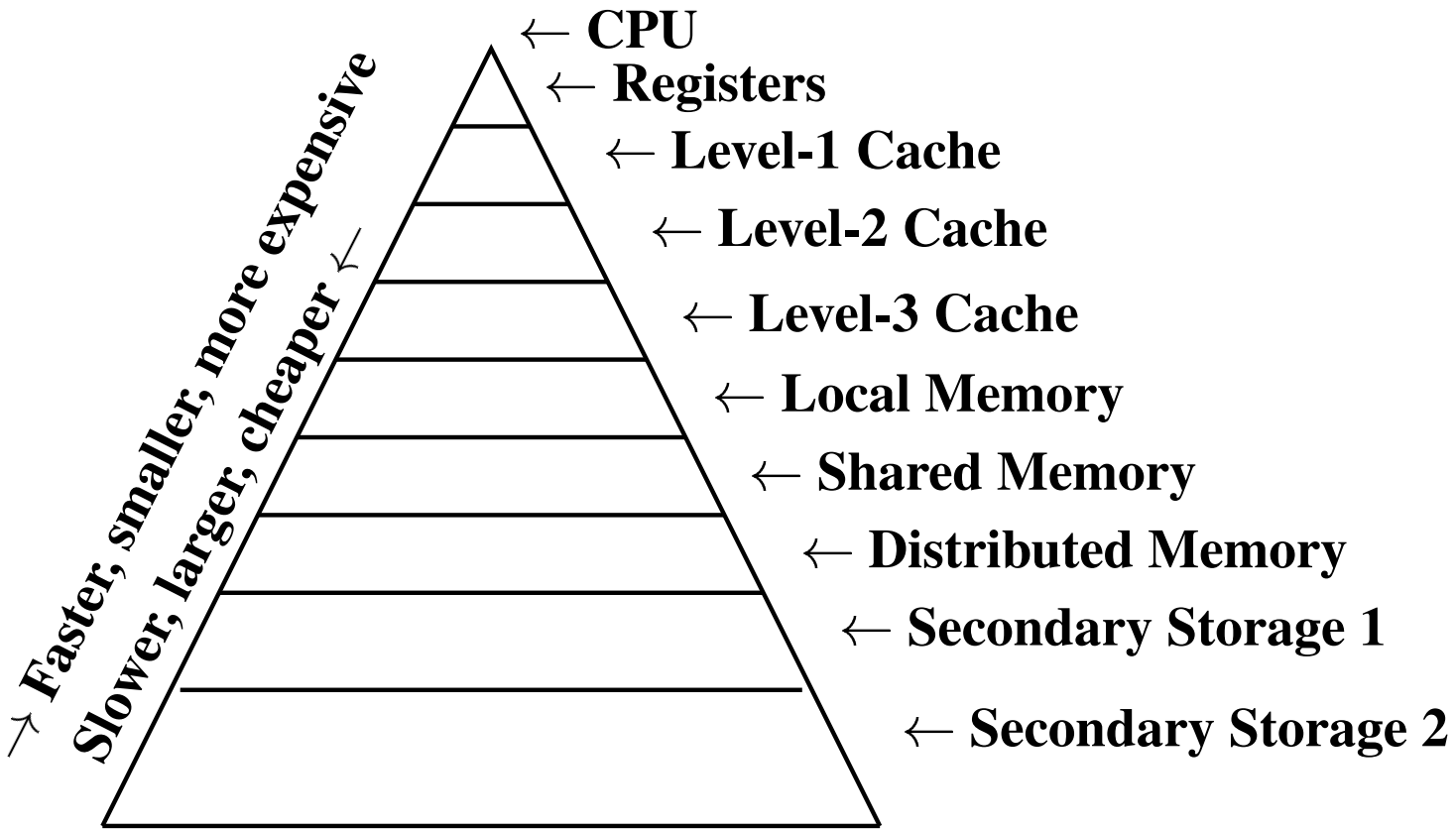
B. S. Andersen, F. G. Gustavson, J. Waśniewski. A recursive formulation of Cholesky factorization of a matrix in packed storage. *TOMS*, Vol. 27, No. 2, June. 2001, pp. 214-244.

Cholesky, a packed and full data storage



B. S. Andersen, F. G. Gustavson, J. Waśniewski. A recursive formulation of Cholesky factorization of a matrix in packed storage. *TOMS*, Vol. 27, No. 2, June. 2001, pp. 214-244.

A computer memory hierarchy



Essential

- **Floating point arithmetic cannot be done unless the operands involved first reside in the L1 (even L0) cache.**
- **Two dimensional Fortran and C arrays do not map nicely into L1 cache.**
  - **The best case happens when the array is contiguous and properly aligned.**
  - **At least three-way set associative cache is required when matrix multiply is being done.**

Mflops, Cholesky Factorizations, lower case, nb = 200, SUN UltraSPARC III.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack	183	247	296	312	321	325	328	331	315	200	169
f. lapack	299	436	637	842	933	1086	864	1173	1203	1169	1236
p. recursive+	95	202	275	426	550	727	913	1010	1093	1118	1215
p. recursive	102	219	290	454	581	760	945	1043	1146	1162	1249
p. hybrid+	390	557	708	782	867	965	1080	1180	1201	1206	1254
p. hybrid	529	778	959	973	1003	1077	1164	1267	1277	1257	1304

B.S. Andersen, J. Gunnar, F.G. Gustavson, J.K. Reid, J. Waśniewski. A fully portable High Performance Minimal Storage Cholesky Algorithm. ACM Trans. on Math. Software 31, 2005, 201-227.

A Minimal Storage Cholesky Algorithm

Mflops, Cholesky factorizations, lower case, nb = 100, IBM Power4.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack	747	951	1043	1024	1059	1101	1037	709	638	621	635
v. p. lapack	1750	2359	2658	2346	3107	3560	3773	3870	3969	3815	3836
f. lapack	440	722	1390	2119	2562	3242	3495	3797	3901	3787	4010
v. f. lapack	1492	2165	2486	3194	3454	3677	3832	3921	4162	4037	4327
p. recursive+	170	379	593	1024	1586	2077	2621	3030	3434	3555	3943
p. recursive	181	406	618	1060	1652	2133	2700	3111	3523	3604	3980
p. hybrid+	878	1488	2085	2721	3211	3754	3974	4112	4188	4200	4275
p. hybrid	1006	1717	2334	2977	3441	3938	4149	4279	4266	4269	4309

A Minimal Storage Cholesky Algorithm

Mflops, Cholesky Factorizations, lower case, nb = 200, HP Itanium 2.

n	40	64	100	160	250	400	640	1000	1600	2500	4000
p. lapack	218	328	463	644	825	952	1039	660	609	595	590
f. lapack	376	581	838	1307	1728	2189	2546	2777	2904	3028	3141
p. recursive+	110	232	378	698	1079	1608	2170	2531	2758	2861	3013
p. recursive	121	256	411	742	1147	1675	2305	2666	2874	2942	3056
p. hybrid+	657	1017	1689	1731	1279	1384	1552	1829	2089	2216	2402
p. hybrid	763	1139	1849	1815	1390	1504	1661	1910	2167	2274	2438

Rectangular Full Packed Data Format, n is odd

$$\mathbf{n} = 7, \qquad \mathbf{memory\ needed} = \mathbf{n} \times \mathbf{n} = 49$$

$$A = \left(\begin{array}{ccccccc}
a_{1,1_1} & \diamond & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{2,1_2} & a_{2,2_9} & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{3,1_3} & a_{3,2_{10}} & a_{3,3_{17}} & \diamond & \diamond & \diamond & \diamond \\
a_{4,1_4} & a_{4,2_{11}} & a_{4,3_{18}} & a_{4,4_{25}} & \diamond & \diamond & \diamond \\
a_{5,1_5} & a_{5,2_{12}} & a_{5,3_{19}} & a_{5,4_{26}} & a_{5,5_{33}} & \diamond & \diamond \\
a_{6,1_6} & a_{6,2_{13}} & a_{6,3_{20}} & a_{6,4_{27}} & a_{6,5_{34}} & a_{6,6_{41}} & \diamond \\
a_{7,1_7} & a_{7,2_{14}} & a_{7,3_{21}} & a_{7,4_{28}} & a_{7,5_{35}} & a_{7,6_{42}} & a_{7,7_{49}}
\end{array}\right)$$

LAPACK full data format

$$\mathbf{n} = 7, \qquad \mathbf{memory\ needed} = \mathbf{n} \times \mathbf{(n+1)/2} = 28$$

$$A^{rfp} = \left(\begin{array}{cccc}
a_{1,1_1} & a_{5,5_8} & a_{6,5_{15}} & a_{7,5_{22}} \\
a_{2,1_2} & a_{2,2_9} & a_{6,6_{16}} & a_{7,6_{23}} \\
a_{3,1_3} & a_{3,2_{10}} & a_{3,3_{17}} & a_{7,7_{24}} \\
a_{4,1_4} & a_{4,2_{11}} & a_{4,3_{18}} & a_{4,4_{25}} \\
a_{5,1_5} & a_{5,2_{12}} & a_{5,3_{19}} & a_{5,4_{26}} \\
a_{6,1_6} & a_{6,2_{13}} & a_{6,3_{20}} & a_{6,4_{27}} \\
a_{7,1_7} & a_{7,2_{14}} & a_{7,3_{21}} & a_{7,4_{28}}
\end{array}\right)$$

Rectangular full packed data format

Rectangular Full Packed Data Format, n is even

$$\mathbf{n} = 6, \qquad \mathbf{memory\ needed} = \mathbf{n} \times \mathbf{n} = 36$$

$$A = \left(\begin{array}{ccc|ccc}
a_{1,1_1} & \diamond & \diamond & \diamond & \diamond & \diamond \\
a_{2,1_2} & a_{2,2_8} & \diamond & \diamond & \diamond & \diamond \\
a_{3,1_3} & a_{3,2_9} & a_{3,3_{15}} & \diamond & \diamond & \diamond \\
a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{16}} & a_{4,4_{22}} & \diamond & \diamond \\
a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{17}} & a_{5,4_{23}} & a_{5,5_{29}} & \diamond \\
a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{18}} & a_{6,4_{24}} & a_{6,5_{30}} & a_{6,6_{36}}
\end{array}\right)$$

LAPACK full data format

$$\mathbf{n} = 6, \qquad \mathbf{memory\ needed} = \mathbf{(n+1)} \times \mathbf{n/2} = 21$$

$$A^{rfp} = \left(\begin{array}{ccc}
a_{4,4_1} & a_{5,4_8} & a_{6,4_{15}} \\
\hline
a_{1,1_2} & a_{5,5_9} & a_{6,5_{16}} \\
a_{2,1_3} & a_{2,2_{10}} & a_{6,6_{17}} \\
\hline
a_{3,1_4} & a_{3,2_{11}} & a_{3,3_{18}} \\
a_{4,1_5} & a_{4,2_{12}} & a_{4,3_{19}} \\
a_{5,1_6} & a_{5,2_{13}} & a_{5,3_{20}} \\
a_{6,1_7} & a_{6,2_{14}} & a_{6,3_{21}}
\end{array}\right)$$

Rectangular full packed data format

Symmetric/Hermitian matrices

$\mathbf{n} = 7, \quad$ **memory needed** $= \mathbf{n} \times \mathbf{(n+1)/2} = 28$

$$A^{rfp} = \begin{pmatrix} a_{1,1_1} & a_{5,5_8} & a_{6,5_{15}} & a_{7,5_{22}} \\ a_{2,1_2} & a_{2,2_9} & a_{6,6_{16}} & a_{7,6_{23}} \\ a_{3,1_3} & a_{3,2_{10}} & a_{3,3_{17}} & a_{7,7_{24}} \\ a_{4,1_4} & a_{4,2_{11}} & a_{4,3_{18}} & a_{4,4_{25}} \\ a_{5,1_5} & a_{5,2_{12}} & a_{5,3_{19}} & a_{5,4_{26}} \\ a_{6,1_6} & a_{6,2_{13}} & a_{6,3_{20}} & a_{6,4_{27}} \\ a_{7,1_7} & a_{7,2_{14}} & a_{7,3_{21}} & a_{7,4_{28}} \end{pmatrix}$$

Rectangular full packed data format

$$A^{rfp} = \begin{pmatrix} A_{11} & A_{22}^T \\ A_{21} & \end{pmatrix}$$

$$L^{rfp} = \begin{pmatrix} L_{11} & L_{22}^T \\ L_{21} & \end{pmatrix} ?$$

- L_{11} := **POTRF of** (A_{11})
- L_{21} := **TRSM of** $(L_{21} L_{11}^T = A_{21})$
- $\hat{L}_{22}$:= **SYRK of** $(A_{22} - L_{21} L_{21}^T)$
- L_{22} := **POTRF of** $(\hat{L}_{22})$

There are eight cases:

1. odd, lower, no transpose
2. odd, lower, transpose
3. odd, upper, no transpose
4. odd, upper, transpose
5. even, lower, no transpose
6. even, lower, transpose
7. even, upper, no transpose
8. even, upper, transpose

Mflops of Cholesky Factorization on SUN UltraSPARC-III

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	456	520	516	482	460	464	291	294
100	753	813	829	768	612	827	399	369
200	946	979	997	955	933	1150	455	370
400	1208	1231	1158	1183	1081	1244	483	339
500	1173	1227	1138	1186	1121	1340	511	343
800	1316	1318	1189	1269	1256	1310	522	324
1000	1275	1318	1281	1303	1313	1406	530	288
1600	1350	1387	1358	1312	1405	1234	502	223
2000	1264	1367	1403	1323	1360	1491	394	163
4000	1287	1450	1537	1263	1392	1565	300	153

Mflops of Cholesky Inversion on SUN UltraSPARC-III

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	379	379	380	381	330	328	318	338
100	698	696	699	700	698	707	412	446
200	1012	989	1008	997	1052	1030	467	558
400	1290	1223	1229	1263	1229	1212	452	606
500	1290	1276	1238	1330	1285	1276	448	595
800	1446	1445	1330	1356	1343	1325	408	566
1000	1428	1337	1442	1436	1378	1318	404	531
1600	1418	1372	1369	1396	1142	1317	262	450
2000	1387	1333	1370	1536	1400	1366	242	394
4000	1460	1408	1389	1453	1421	1395	201	288

Mflops of Cholesky Solution on SUN UltraSPARC-III

		rfp				po		lapack	
		no trans		trans				pp	
nrhs	n	u	l	u	l	u	l	u	l
100	50	1132	1123	1153	1135	1103	1069	353	353
100	100	1193	1163	1237	1211	1262	1262	478	478
100	200	1477	1478	1500	1490	1280	1235	557	554
100	400	1494	1505	1514	1534	1150	1149	582	582
100	500	1466	1443	1436	1445	1217	1229	560	569
100	800	1503	1505	1535	1469	1151	1096	528	526
100	1000	1553	1524	1499	1576	1089	1125	513	513
160	1600	1595	1564	1603	1577	1155	1121	421	403
200	2000	1600	1636	1610	1615	1105	1087	347	338
400	4000	1666	1668	1696	1665	1080	1084	292	290

ia64 Itanium (DMI) @ 1300 MHz

Cholesky: factorization, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	781	771	784	771	1107	739	495	533
100	1843	1788	1848	1812	1874	1725	879	825
200	3178	2869	2963	3064	2967	2871	1323	1100
400	3931	3709	3756	3823	3870	3740	1121	1236
500	4008	3808	3883	3914	4043	3911	1032	1257
800	4198	4097	4145	4126	3900	4009	612	1127
1000	4115	4038	4015	3649	3769	3983	305	697
1600	3851	3652	3967	3971	3640	3987	147	437
2000	3899	3716	3660	3660	3865	3835	108	358
4000	3966	3791	3927	4011	3869	4052	119	398

ia64 Itanium (DMI) @ 1300 MHz

Cholesky: inversion, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	633	659	648	640	777	870	508	460
100	1252	1323	1300	1272	1573	1760	815	810
200	2305	2442	2431	2314	2357	2639	1118	1211
400	3084	3199	3188	3094	3152	3445	1234	1363
500	3204	3316	3329	3218	3400	3611	1239	1382
800	3617	3741	3720	3640	3468	3786	1182	1268
1000	3611	3716	3637	3590	3456	3790	767	946
1600	3721	3802	3795	3714	3589	3713	500	609
2000	3784	3812	3745	3704	3636	3798	473	596
4000	3822	3762	3956	3851	3760	3750	467	614

ia64 Itanium (DMI) @ 1300 MHz

Cholesky: solution, NEC BLAS library, double prec.

		rfp				lapack			
		no trans		trans		po		pp	
nrhs	n	u	l	u	l	u	l	u	l
100	50	2409	2412	2414	2422	3044	3018	725	714
100	100	3305	3301	3303	3303	3889	3855	1126	1109
100	200	4149	4154	4127	4146	4143	4127	1526	1512
100	400	4398	4403	4416	4444	4469	4451	1097	1088
100	500	4313	4155	4374	4394	4203	4093	1054	1045
100	800	3979	3919	4040	4051	3969	4011	692	720
100	1000	3716	3608	3498	3477	3630	3645	376	372
160	1600	3892	3874	4020	3994	4001	4011	188	182
200	2000	4052	4073	4040	4020	4231	4203	119	119
400	4000	4245	4225	4275	4287	4330	4320	115	144

SX-6 NEC (DMI) @ ... MHz, Vector option

Cholesky: factorization, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	206	200	225	225	365	353	57	238
100	721	728	789	788	1055	989	120	591
200	2028	2025	2005	2015	1380	1639	246	1250
400	3868	3915	3078	3073	1763	3311	479	1975
500	4483	4470	4636	4636	4103	4241	585	2149
800	5154	5168	4331	4261	3253	4469	870	2399
1000	5666	5654	5725	5703	5144	5689	1035	2474
1600	6224	6145	5644	5272	5375	5895	1441	2572
2000	6762	6788	6642	6610	6088	6732	1654	2598
4000	7321	7325	7236	7125	6994	7311	2339	2641

SX-6 NEC (DMI) @ ... MHz, Vector option

Cholesky: inversion, NEC BLAS library, double prec.

	rfp				lapack			
	no trans		trans		po		pp	
n	u	l	u	l	u	l	u	l
50	152	152	150	152	148	145	91	61
100	430	432	428	432	313	310	194	126
200	950	956	940	941	636	627	404	249
400	1850	1852	1804	1806	1734	1624	722	470
500	2227	2228	2174	2181	2180	2029	856	572
800	3775	3775	3668	3686	3405	3052	1186	842
1000	4346	4346	4254	4263	4273	3638	1342	985
1600	5313	5294	5137	5308	5438	4511	1690	1361
2000	6006	6006	5930	5931	5997	4832	1854	1536
4000	6953	6953	6836	6888	7041	4814	1921	2122

IBM Power4 computer

n	n proc	Mflp rfptrf	Times					
			in rfptrf				lapack	
			potrf	trsm	syrk	potrf	potrf	pptrf
1000	1	2695	0.12	0.02	0.05	0.04	0.02	0.12	0.94
	5	7570	0.04	0.01	0.02	0.01	0.01	0.03	0.32
	10	10699	0.03	0.01	0.01	0.01	0.00	0.02	0.16
	15	9114	0.04	0.01	0.02	0.01	0.01	0.02	0.16
2000	1	2618	1.02	0.13	0.38	0.38	0.13	0.97	8.74
	5	10127	0.26	0.04	0.10	0.09	0.04	0.24	3.42
	10	17579	0.15	0.02	0.06	0.05	0.03	0.12	1.65
	15	23798	0.11	0.02	0.04	0.04	0.01	0.13	1.11
3000	1	2577	3.49	0.45	1.33	1.28	0.44	3.40	30.42
	5	11369	0.79	0.11	0.28	0.30	0.11	0.71	11.76
	10	19706	0.46	0.06	0.19	0.16	0.05	0.38	6.16
	15	29280	0.31	0.05	0.12	0.10	0.04	0.26	4.28
4000	1	2664	8.01	1.01	2.90	3.09	1.01	7.55	75.72
	5	11221	1.90	0.26	0.68	0.72	0.24	1.65	25.73
	10	21275	1.00	0.13	0.39	0.36	0.12	0.86	13.95
	15	31024	0.69	0.09	0.28	0.24	0.08	0.59	10.46
5000	1	2551	16.34	2.04	6.16	6.10	2.04	15.79	154.74
	5	11372	3.66	0.45	1.37	1.44	0.40	3.27	47.76
	10	22326	1.87	0.25	0.78	0.62	0.22	1.73	28.13
	15	32265	1.29	0.17	0.53	0.45	0.14	1.16	20.95

SUN UltraSPARC-IV computer

n	n proc	Mflp rfptrf	Times	Times in rfptrf				Times lapack	
				potrf	trsm	syrk	potrf	potrf	pptrf
1000	1	1587	0.21	0.03	0.09	0.07	0.03	0.19	1.06
	5	4762	0.07	0.02	0.02	0.02	0.02	0.07	1.13
	10	5557	0.06	0.01	0.01	0.02	0.02	0.06	1.12
	15	5557	0.06	0.02	0.01	0.01	0.02	0.06	1.11
2000	1	1668	1.58	0.22	0.63	0.52	0.22	1.45	11.20
	5	6667	0.40	0.07	0.13	0.13	0.07	0.38	11.95
	10	8602	0.31	0.06	0.07	0.11	0.07	0.25	11.24
	15	9524	0.28	0.06	0.06	0.08	0.08	0.23	11.66
3000	1	1819	4.95	0.62	1.98	1.72	0.63	4.86	45.48
	5	6872	1.31	0.20	0.42	0.48	0.20	1.38	55.77
	10	12162	0.74	0.14	0.22	0.21	0.16	0.76	46.99
	15	12676	0.71	0.14	0.16	0.30	0.16	0.61	45.71
4000	1	1823	11.70	1.52	4.62	4.01	1.55	11.86	112.52
	5	7960	2.68	0.40	0.94	0.92	0.42	2.74	112.77
	10	14035	1.52	0.26	0.47	0.49	0.30	1.61	112.53
	15	17067	1.25	0.24	0.37	0.35	0.29	1.29	111.67
5000	1	1843	22.61	2.92	8.76	8.00	2.93	23.60	218.94
	5	8139	5.12	0.77	1.81	1.80	0.74	5.45	221.58
	10	14318	2.91	0.50	0.97	0.93	0.51	3.11	214.54
	15	17960	2.32	0.45	0.72	0.68	0.47	2.40	225.08

Conclusions:

1. **The Rectangular Full Packed (RFP) data format can replace both LAPACK, full and packed, data formats for symmetric and triangular matrices. POTRF or POTRI are still as an auxiliary routine.**
2. **RFP saves almost half $n(n-1)/2$ space comparing with LAPACK full data format but it runs with the same speed.**
3. **Of course the speed of RFP data format can be still improved.**
4. **RFP is many times faster than LAPACK packed data format**
5. **RFP Algorithm and routines for the distributed memory computers (ScaLAPACK) have already been developed by Fred Gustavson with his Umea (Sweden) colleagues**

Data Formats for Dense Linear Algebra Algorithms speeds up the computation and reduce the data storage with a historical introduction

Fred Gustavson[1] and Jerzy Waśniewski[2](speaker)

1) IBM Research Center, Yorktown Height

e-mail: fg2@us.ibm.com

2) Technical University of Denmark

e-mail: jw@imm.dtu.dk

Slides:

http://www.imm.dtu.dk/~jw/Lectures/pmaa06.pdf

http://www.imm.dtu.dk/~jw/Lectures/pmaa06.ps.gz

August 29, 2006