A Critical Study of Efficient Multi-core EM Clustering

D. J. Nagendra Kumar
BVRICE
Bhimavaram
AP, India

J. V. R. Murthy
JNTUK
Kakinada
AP, India

N. B. Venkateswarlu
AITAM
Tekkali
AP, India

ABSTRACT
The state-of-the-art computer hardware is coming with multi-core processors. Even mobile phones are coming with dual-core processors. OpenMP is one technology supporting parallel programming on multi-core shared memory systems with the help of threads. In this paper, we observed the execution times Serial EM Clustering running on single-core and Parallel EM Clustering methods using OpenMP on I3 system. Observations are made varying number of threads, samples, dimensions and clusters. The results show that OpenMP Lower Triangular Canonical Form with Forward Substitution and Winograd's approach (OLFW) EM gives a considerable speed-up of 2.7 over serial standard EM.

General Terms
Performance, Experimentation and Verification.

Keywords
OpenMP, Multi-core, Parallel programming, Expectation Maximization, and EM.

1. INTRODUCTION
Manasi N. Joshi [1] describes clustering large datasets as time consuming and processor intensive. S. N. Tirumala Rao et al. [2] experimented with parallel k-means using OpenMP and Posix threads on shared-memory multi-core systems, and found that OpenMP k-means clustering works faster than both single-core k-means and multi-core Posix threads k-means clustering. Similarly [3][4][5] are some OpenMP studies from Data Mining community. EM clustering algorithm is the second dominantly used clustering algorithm, next to k-means [6]. In continuation to our search of devising faster EM algorithms [7][8][9], here we experimented with OpenMP implementation of various EM approaches from [8][9].

2. OPENMP
OpenMP is a shared memory application programming interface (API). It is not a new programming language. It is just a notation that can be added to a program in C, or C++, or Fortran, describing how the work is to be shared among the threads executing on multi-core processors sharing data in shared memory. Posix is another approach for the same purpose, but Posix thread programming is a little bit tougher.

In majority of our programs, there will be implicit parallelism, in the form of loops or tasks. Loop parallelism in OpenMP is easy to code with just one line of OpenMP directive. Of course in the parallel loop directive one can specify the data to be shared, the data to be kept private to the thread, the maximum number of threads to be used, and etc.

3. PARALLEL EM CLUSTERING
3.1 EM Clustering
Expectation maximization (EM) is a widely used mixture model-based clustering algorithm proposed by Dempster et. al [10]. EM clustering groups the given data samples into k Gaussian-distributions probabilistically. The outline of EM is as follows:

1. Select the number of clusters, k.
2. Initialize cluster parameters (cluster weights, cluster means, and covariance matrices).
3. Perform Expectation step.
   Using probability density function of normal distribution, find probability of each sample belonging to each of the k clusters.
4. Perform Maximization step.
   Compute the new cluster weights, cluster means and covariance matrices of all clusters.
5. Go to step 3 until either the log-likelihood value is considerably small or the maximum number of iterations are exhausted.

3.2 Parallel OpenMP implementation of EM
The steps 3 and 4 above involve complete scans of data. Here is where we can parallelize the loops. The pseudo code for parallel EM can be given as:

1. Select the number of clusters, k.
2. Initialize cluster parameters (cluster weights, cluster means, and covariance matrices) and set number of threads.
3. Perform Expectation step.
   #pragma omp parallel for
   Using probability density function of normal distribution, find probability of each sample belonging to each of the k clusters.
4. Perform Maximization step.
   #pragma omp parallel for
   Compute the new cluster weights, cluster means and covariance matrices of all clusters.
5. Go to step 3 until either the log-likelihood value is considerably small or the maximum number of iterations are exhausted.

3.3 Parallel OpenMP-EM approaches
There are 9 approaches possible to fast-up the quadratic term computation in Expectation step of EM algorithm (step 3 of pseudo code in section 3.2) from [5][6]:

Method 1. OpenMP Parallel implementation of Standard EM (OSEM)
Method 2. OpenMP Parallel implementation of Lower Triangular Canonical Form with Matrix Inversion (OLTI)
Method 3. OpenMP Parallel implementation of Lower Triangular Form with Forward Substitution (OLTF)
Method 4. OpenMP Parallel implementation of Cascaded approach (OCAS)
Method 5. OpenMP Parallel implementation of EM with Winograd's method for vector-matrix multiplication (OEMW)
Method 6. OpenMP Parallel implementation of Lower Triangular Canonical Form with Matrix Inversion and Winograd's approach (OLIW)

Method 7. OpenMP Parallel implementation of Lower Triangular Canonical Form with Forward Substitution and Winograd's approach (OLFW)
Method 8. OpenMP Parallel implementation of Cascaded approach with Winograd's approach (OCAW)
Method 9. OpenMP Parallel implementation of Unitary Canonical Form with Winograd's approach (OUFW)

4. EXPERIMENTATION AND RESULTS

A system with Core i3 processor, 4GB RAM, Fedora 64-bit Linux 12.04 Operating System is used for experimentation. GNU GCC compiler for C language is used. The following Experimentation is taken up on synthetic data:

1. Parallel OpenMP implementations of EM with 4-threads on synthetic data of 1 Million rows, 50 dimensions and clusters varying from 5 to 10.
2. Parallel OpenMP EM implementations with 4-threads on synthetic data of 1 Million rows, 5 clusters and dimensions varying from 50 to 90.
3. Parallel OpenMP EM with 4-threads on synthetic data of 50 dimensions and 5 clusters and number of samples varying from 0.5 Million to 2.5 Millions.
4. Parallel OpenMP EM on synthetic data of 1 Million samples, 50 dimensions and 5 clusters, where the number of threads are from 2 to 10.

In all the above cases, the timings of serial single-core EM are also observed and used for comparative purposes. The speed-ups of Parallel EM versions compared to that of serial single-core EM are noted down.

4.1 Varying Number of Clusters

The observations of parallel EM on a synthetic dataset of 1 Million samples, 50 dimensions with 4-threads and clusters from 5 to 10 are analyzed here. Table 1 and Fig. 1 gives the timing observations and the speed-ups.

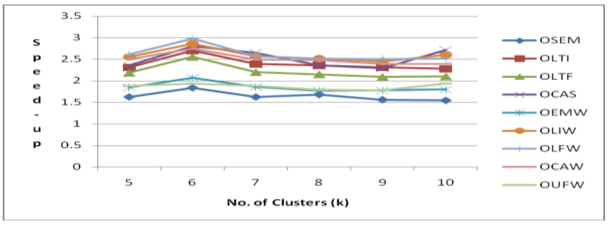

Fig. 1: Speed-up of Parallel EM compared to single-core serial EM, SEM. The average speed-up of OLFW, the best of all the proposed methods, is 2.61.

4.2 Varying Number of Dimensions

The observations of parallel EM on synthetic dataset of 1 Million samples and 5 clusters with 4-threads and number of dimensions changing from 50 to 90 are analyzed here. Table 2 and Fig. 2 give the timing observations and their speed-ups.

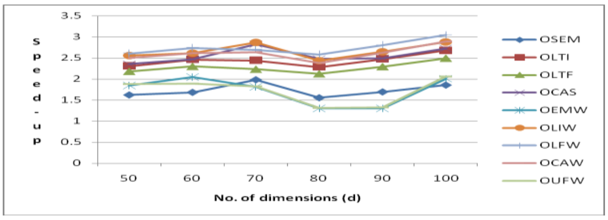

Fig. 2: Speed-up of Parallel EM compared to single-core SEM. Number of dimensions varies from 50 to 90. The best of all average speed-ups achieved is, that of OLFW EM, 2.75.

4.3 Varying Number of Samples

The observations of parallel EM on synthetic dataset of 50 dimensions and 5 clusters with 4-threads and number of samples ranging from 0.5 Million rows to 2.5 Million rows are analyzed here. Table 3 and Fig. 3 give the timing observations and their speed-ups.

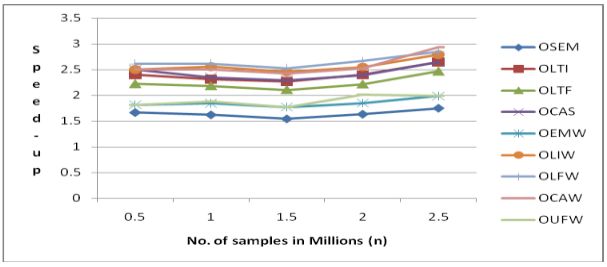

Fig. 3: Speed-up of Parallel EM compared to single-core EM. Number of Samples ranges from 0.5 Million to 2.5 Million rows. The average speed-up of OLFW EM is 2.65.

4.4 Varying Number of Threads

The observations of parallel EM on synthetic data with 1 Million rows, 50 dimensions, 5 clusters and number of threads from 2 to 10 are analyzed here. Table 4, Table 5 and Fig. 4 give the timing observations and their speed-ups.

Here we observed real time and system time taken by each experiment. The real time (elapsed time) is the time gap between the invocation of the program and termination of the program. This is time the user generally observes. The system time is the amount of time the CPU spends in kernel mode. System time is calculated over all the multiple cores available. Since the CPU utilizes all its cores for processing parallel EM, system time is more than real time in parallel EM. However in case of Standard EM, system is less than real time, as it uses a only one core.

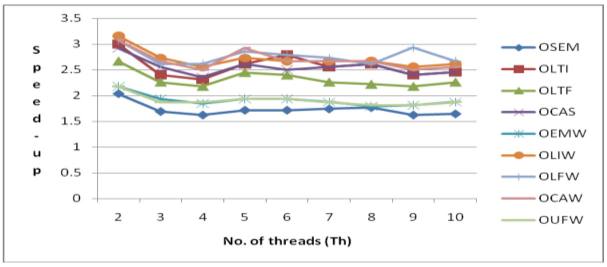

Fig. 4: Speed-up of Parallel EM compared to single-core serial EM (SEM). Number of Threads ranges from 2 to 10. The average speed-up of OLFW EM is 2.76.

Observing Table 4 and Table 5, one can deduct that from 3 threads onwards there is no much improvement in speed-up. There are only two cores in our system. Hence in a system with n-cores in processor, there won't be much improvement beyond n-threads of OpenMP.

5. CONCLUSION

OpenMP based multi-core EM clustering is running much faster compared to single-core EM. In this work, we observed the execution times of all the above EM clustering methods using OpenMP running on Intel Core I3 system (with 2-cores and 4-hyper threads), varying number of threads (Th), samples (n), dimensions (d), and clusters (k). The performance of the proposed openMP EM approaches (methods 1 to 9) is compared to that of the standard EM, and found that OLFW EM is the best of all the proposed approaches. The results show that OLFW gives a considerable speed-up of 2.7 over standard (serial) EM.

6. REFERENCES

[1] Manasi N. Joshi, "Parallel K-means Algorithm on Distributed Memory Multiprocessors", Project Report, Computer Science Department, University of Minnesota, Twin Cities, Spring 2003.

[2] S. N. Tirumala Rao, E. V. Prasad, and N. B. Venkateswrlu, "A Critical Performance Study of Memory Mapping on Multi-core Processors: An Experiment with K-means Algorithm with Large Data Mining Data Sets", IJCA (0975-8887) 2010 Volume 1- No. 9.

[3] M.D. Jones, R. Yao, and C.P. Bhole, "Hybrid MPI-OpenMP Programming for Parallel OSEM PET Reconstruction", IEEE Trans. On Nuclear Science, Vol. 53, No. 5, October 2006, pp. 2752-2758.

[4] Jones M.D. and Yao R., "Parallel Programming for OSEM reconstruction with MPI, OpenMP, and hybrid MPI-OpenMP", IEEE Nuclear Science Symposium Conferecience Record, 2004, Vol. 5, pp. 3036-3042.

[5] Waghmare, Vivek N. and Kulkarni, Dinesh B., "Convex Hull Using K-means Clustering in Hybrid (MPI/OpenMP) Environment", Int. Conf. on Computational Intelligence and Communication Networks 2010, pp. 150-153.

[6] Ian H. Witten, Eibe Frank, and Mark A. Hall. Data Mining. Third Edition. Morgan Kaufmann.

[7] D. J. Nagendra Kumar and J. V. R. Murthy, "Some Studies of Expectation Maximization Clustering Algorithm to Enhance Performance", Research Cell: An International Journal of Engineering Sciences, ISSN: 2229-6913, Vol. 2, July 2011, pp. 254-269.

[8] D. J. Nagendra Kumar, J. V. R. Murthy and N. B. Venkateswarlu, "Fast Expectation Maximization Clustering Algorithm", International Journal of Computational Intelligence Research (IJCIR) ISSN: 0973-1873, Vol. 8, No. 2 (2012), pp. 71-94.

[9] D. J. Nagendra Kumar, J. V. R. Murthy and N. B. Venkateswarlu, "Computation Reduction of Expectation Maximization Clustering Using Winograd's Method", Proceedings of IEEE International Conference on Electronics Computer Technology ICECT-2012, pp. 255-259.

[10] A. P. Dempster and N. M. Laird and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. Journal of The Royal Statistical Society, Series B, 39(1):1--38, 1977.

Table 1. Timing observations (in sec) of Standard EM and OpenMP EM approaches with 4 threads on Synthetic dataset of 1 Million rows (n), 50 dimensions (d), and varying number of clusters (k) 5 to 10

		Execution Time									Speed-up								
k	SEM	OSEM	OLTI	OLTF	OCAS	OEMW	OLIW	OLFW	OCAW	OUFW	OSEM	OLTI	OLTF	OCAS	OEMW	OLIW	OLFW	OCAW	OUFW
5	120	74	52	55	51	65	47	46	48	64	1.62	2.31	2.18	2.35	1.85	2.55	2.61	2.50	1.88
6	170	78	53	56	51	69	50	48	52	74	1.83	2.70	2.55	2.80	2.07	2.86	2.98	2.75	1.93
7	232	103	70	76	63	90	65	65	67	89	1.62	2.39	2.20	2.65	1.86	2.57	2.57	2.49	1.88
8	297	114	81	89	81	108	76	76	77	107	1.68	2.36	2.15	2.36	1.77	2.51	2.51	2.48	1.79
9	376	138	93	103	94	121	88	86	90	121	1.56	2.31	2.09	2.29	1.78	2.44	2.50	2.39	1.78
10	466	155	105	114	88	133	92	95	100	124	1.54	2.28	2.10	2.72	1.80	2.60	2.52	2.39	1.93
Avg.											**1.64**	**2.39**	**2.21**	**2.53**	**1.85**	**2.59**	**2.61**	**2.50**	**1.86**

Table 2. Timing observations (in sec) of Standard EM and OpenMP EM approaches with 4 threads on Synthetic dataset of 1 Million rows (n), 5 clusters (k), and varying number of dimensions (d) from 50 to 90

		Execution Time									Speed-up								
k	SEM	OSEM	OLTI	OLTF	OCAS	OEMW	OLIW	OLFW	OCAW	OUFW	OSEM	OLTI	OLTF	OCAS	OEMW	OLIW	OLFW	OCAW	OUFW
50	120	74	52	55	51	65	47	46	48	64	1.62	2.31	2.18	2.35	1.85	2.55	2.61	2.50	1.88
60	170	101	69	74	69	83	65	62	65	90	1.68	2.46	2.30	2.46	2.05	2.62	2.74	2.62	1.89
70	232	117	95	104	82	128	81	86	88	127	1.98	2.44	2.23	2.83	1.81	2.86	2.70	2.64	1.83
80	297	191	130	140	120	229	122	115	125	226	1.55	2.28	2.12	2.48	1.30	2.43	2.58	2.38	1.31
90	376	222	152	164	151	290	142	134	143	283	1.69	2.47	2.29	2.49	1.30	2.65	2.81	2.63	1.33
100	466	251	174	187	171	232	162	153	161	227	1.86	2.68	2.49	2.73	2.01	2.88	3.05	2.89	2.05
Avg.											**1.73**	**2.44**	**2.27**	**2.56**	**1.72**	**2.67**	**2.75**	**2.61**	**1.71**

Table 3. Timing observations (in sec) of Standard EM and OpenMP EM approaches with 4 threads on Synthetic dataset with 50 dimensions (d), 5 clusters (k), and varying number of rows (n) from 0.5 Million to 2.5 Million

		Execution Time									Speed-up								
k	SEM	OSEM	OLTI	OLTF	OCAS	OEMW	OLIW	OLFW	OCAW	OUFW	OSEM	OLTI	OLTF	OCAS	OEMW	OLIW	OLFW	OCAW	OUFW
1	36	36	25	27	24	33	24	23	24	33	1.67	2.40	2.22	2.50	1.82	2.50	2.61	2.50	1.82
1	74	74	52	55	51	65	47	46	48	64	1.62	2.31	2.18	2.35	1.85	2.55	2.61	2.50	1.88
2	116	116	79	85	78	101	73	71	74	101	1.54	2.27	2.11	2.29	1.77	2.45	2.52	2.42	1.77
2	145	145	99	107	99	128	93	89	94	118	1.63	2.39	2.21	2.39	1.85	2.55	2.66	2.52	2.01
3	169	169	112	120	112	149	106	104	101	149	1.75	2.64	2.47	2.64	1.99	2.79	2.85	2.93	1.99
Avg.											**1.64**	**2.40**	**2.24**	**2.44**	**1.85**	**2.57**	**2.65**	**2.57**	**1.89**

Table 4. Execution Times (in sec) of Standard EM and OpenMP EM approaches varying number of threads from 2 to 10 on Synthetic dataset of 1 Million rows (n), 50 dimensions (d), and 5 clusters (k)

			Execution Time																		
	SEM		OSEM		OLTI		OLTF		OCAS		OEMW		OLIW		OLFW		OCAW		OUFW	
Th	Real Time	User Time	Real Time	User Time	Real Time	User Time	Real Time	User Time	Real Time	User Time	Real Time	User Time	Real Time	User Time	Real Time	User Time	Real Time	User Time	Real Time	User Time
1	120	120																		
2			59	117	40	80	45	89	41	80	55	109	38	75	39	76	39	76	55	109
3			71	188	50	131	53	145	47	124	62	165	44	123	46	124	45	121	64	170
4			74	292	52	201	55	215	51	196	65	256	47	186	46	181	48	190	64	254
5			70	273	46	176	49	190	46	177	62	243	44	167	42	163	41	159	62	242
6			70	277	43	168	50	195	48	185	62	243	45	174	43	166	45	174	62	244
7			69	269	47	185	53	207	47	184	64	251	45	177	44	174	45	178	64	251
8			68	269	46	180	54	216	46	185	67	265	45	180	46	183	45	178	66	262
9			74	294	50	200	55	214	50	199	66	259	47	187	41	163	48	199	66	261
10			73	289	49	196	53	210	49	194	64	251	46	183	45	177	47	184	64	254

Table 5. Speed-up of OpenMP EM approaches compared to SEM, varying number of threads from 2 to 10 on Synthetic dataset of 1 Million rows (n), 50 dimensions (d), and 5 clusters (k)

	Speed-up								
Th	OSEM	OLTI	OLTF	OCAS	OEMW	OLIW	OLFW	OCAW	OUFW
2	2.03	3.00	2.67	2.93	2.18	3.16	3.08	3.08	2.18
3	1.69	2.40	2.26	2.55	1.94	2.73	2.61	2.67	1.88
4	1.62	2.31	2.18	2.35	1.85	2.55	2.61	2.50	1.88
5	1.71	2.61	2.45	2.61	1.94	2.73	2.86	2.93	1.94
6	1.71	2.79	2.40	2.50	1.94	2.67	2.79	2.67	1.94
7	1.74	2.55	2.26	2.55	1.88	2.67	2.73	2.67	1.88
8	1.76	2.61	2.22	2.61	1.79	2.67	2.61	2.67	1.82
9	1.62	2.40	2.18	2.40	1.82	2.55	2.93	2.50	1.82
10	1.64	2.45	2.26	2.45	1.88	2.61	2.67	2.55	1.88
Avg.	**1.73**	**2.57**	**2.32**	**2.55**	**1.91**	**2.70**	**2.76**	**2.69**	**1.91**