Performance Evaluation of a Parallel Iterative Method Library using OpenMP

H. Kotakemori [1], H. Hasegawa [2], A. Nishida [1]

[1] University of Tokyo / CREST, JST
[2] University of Tsukuba / CREST, JST

Outline

- Introduction
- Sparse Matrix-Vector Product
- Experiments
  - Sparse matrix-vector product
  - Conversion costs
- Conclusions

Introduction

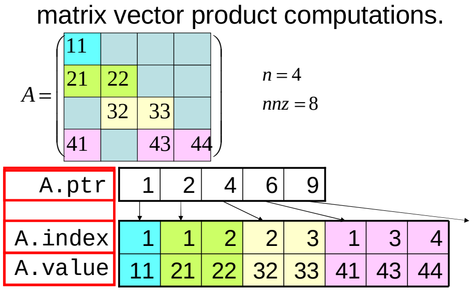
- We are developing Lis (a Library of Iterative Solvers for linear systems), which includes a wide range of iterative solvers, preconditioners, and storage formats.
  - http://ssi.is.s.u-tokyo.ac.jp/lis/
- Performance of Iterative solvers depends on matrix-vector product.
- Number of Iteration does not depend on storage format.
- Fast storage format is essential for iterative method.
- We discuss the performance of sparse matrix-vector products on several shared memory parallel machines.

Sparse Matrix-Vector Product with OpenMP

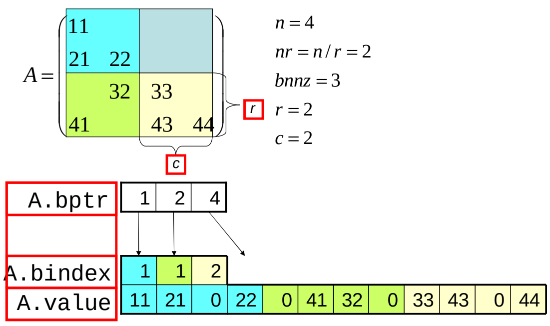
- Sparse Matrix-Vector Product y=Ax
  - The storage formats affect the performance
  - We consider three storage formats (CRS,BSR,DIA)
- Parallelize using OpenMP.
  - OpenMP is designed for shared memory machines.
- Advantages for OpenMP
  - a serial program can be parallelized one loop at a time.
  - Compiler directives are used, so that the same code can be compiled for serial or parallel execution.

Compressed Row Storage (CRS)

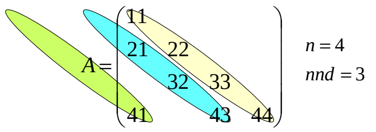
- CRS format is commonly used for sparse matrix vector product computations.



Block Sparse Row (BSR)

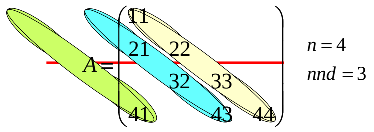


Diagonal (DIA)



A.index	-3	-1	0									
A.value	0	0	0	41	0	21	32	43	11	22	33	44

Modification of DIA



A.index	-3	-1	0									
A.value	0	0	0	41	0	21	32	43	11	22	33	44

⇩

A.index	-3	-1	0									
A.value	0	0	0	21	11	22	0	41	32	43	33	44

Experiments

- Goals
  - Scalability of matrix-vector product
  - Performance of each storage format in each platform
- We examined
  - times of parallel matrix-vector products
  - speed-ups of parallel matrix-vector products
  - storage format conversion costs
- BSR and DIA format is converted from based format CRS

Evaluation platforms

Machine	SGI Altix3700	IBM eServer p5 595	Sun SunFire15K
CPU	Itanium2 1.3GHz	Power5 1.9GHz	USPARCIII+ 900MHz
L1 Cache	16KB	32KB	64KB
L2 Cache	256KB	1.92MB	8MB
L3 Cache	3MB	36MB	---
# of PE	32	64(used 16)	72(used 32)
Memory	32GB	256GB	288GB
OS	Linux	AIX 5L	Solaris 9
Compiler	Intel C/C++8.1	IBM XL C/C++7.0	Sun WorkShop6
Options	-O3	-O3	-O5

Test Matrices

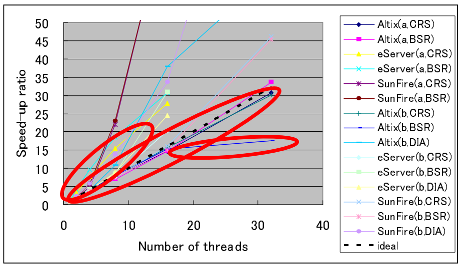
	(a) Matrix Market	(b) FEM of the three-dimensional Poisson equation on a cube
Name	s3dkq4m2	Poisson
Dimension	90,449	1,000,000
# of nonzeros	4,820,891	26,463,592
# of nonzeros per row	53.30	26.46
Memory for CRS	55.5MB	306.7MB
Memory for BSR_21	48.7MB	337.3MB
Memory for BSR_22	46.2MB	453.1MB
Memory for BSR_31	50.2MB	394.9MB
Memory for BSR_41	48.4MB	452.2MB
Memory for DIA	456.1MB	206.0MB

Execution times (in seconds) of 1000 iterations

Number of threads		Format	1	2	4	8	16	32
a	Altix	CRS	20.8	10.47	5.26	2.71	1.43	0.68
		BSR_41	**9.17**	**4.65**	**2.39**	**1.30**	**0.62**	**0.27**
	eServer	CRS	24.11	7.32	3.16	1.56	0.87	
		BSR_41	**18.89**	**4.42**	**1.97**	**1.02**	**0.62**	
	SunFire	CRS	428.13	212.39	87.16	19.46	5.32	2.52
		BSR_22	**348.54**	**168.63**	**60.67**	**15.13**	**4.34**	**1.77**
b	Altix	CRS	149.50	74.96	37.43	18.76	9.51	4.97
		BSR_31	**85.60**	**43.25**	**21.53**	**10.92**	5.63	4.87
		DIA	178.50	89.19	44.34	16.40	**4.72**	**2.81**
	eServer	CRS	154.50	79.63	40.61	20.72	7.62	
		BSR_21	156.27	78.83	40.78	18.57	**5.06**	
		DIA	**147.04**	**71.85**	**34.92**	**16.51**	6.02	
	SunFire	CRS	**2542.74**	**1263.50**	**650.93**	**337.33**	159.38	55.01
		BSR_21	2666.34	1363.94	692.93	353.03	176.88	58.84
		DIA	4523.94	1905.25	791.90	329.20	**134.96**	**35.87**

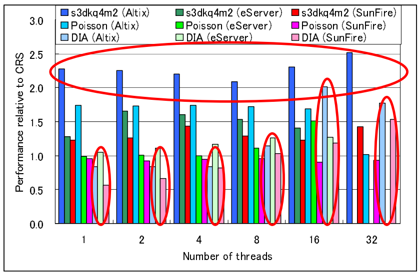
- For the BSR format, the best block size differs for different matrices and machines

Speed-up ratios



- The parallelization speedups on all machines are nearly ideal in most cases.
- DIA format has super-linear speedups from a small number of threads compared with other storage formats.

Performance relative to CRS



- The Altix has a higher relative performance for BSR compared with other machines.

Conversion Costs

- Assumptions:
  - T_{crs} : the execution times of MV in the CRS formats.
  - T_{tgt} : the execution times of MV target formats.
  - T_{conv} : the execution times of the conversion from the CRS format to the target format.
- Conversion Costs

$$N_{th} = \left\lceil \frac{T_{conv}}{T_{crs} - T_{tgt}} \right\rceil$$

  - If the number of MV ≧ Nth then it is better to use the target format; otherwise it is better to use CRS format without conversion.

Threshold numbers of iterations N_{th}

Number of threads		Format	1	2	4	8	16	32
a	Altix	BSR_41	50	51	52	56	60	132
a	eServer	BSR_41	56	51	60	67	75	
a	SunFire	BSR_22	20	18	15	46	112	110
b	Altix	BSR_31	53	55	68	61	79	4306
b	Altix	DIA				76	35	83
b	eServer	BSR_21		1082		104	45	
b	eServer	DIA	93	46	33	24	34	
b	SunFire	BSR_21						
b	SunFire	DIA				87	17	13

- For BSR, the value of Nth increases slightly and is approximately 60 times greater in the Altix and the eServer.
- In the SunFire, the value of Nth increases greatly for eight threads or more.

Conclusions

- Our Implementations have attained satisfactory scalability.
- The storage format has been observed to greatly affect the performance of matrix-vector products.
  - Altix has a higher performance for BSR in this experiments.
  - DIA format has a higher performance, if data is installed on the cache.
- The conversion of the storage format provides faster computation of the matrix-vector product.

Future Works

- Our next goal is parallelization for distributed memory parallel machines through MPI and MPI-OpenMP hybrid parallelization.
- We will also work toward high-performance iterative linear solvers using these kernel routines and effective preconditioners for the solvers.

Acknowledgements

- This research was supported in part by CREST “Development of Software Infrastructure for Large Scale Scientific Simulation”, Japan Science and Technology Agency.