APPENDIX A. *ANALYSES UNDER FERMI PLATFORM*

This appendix contains the results of using auto-tuner for Fermi platform. Results show that our adaptive scheme improves the performance of sparse matrix multiplications by 1.8× for single-precision and 1.4× for double-precision formats.

A.1 Experiments to Set Parameters

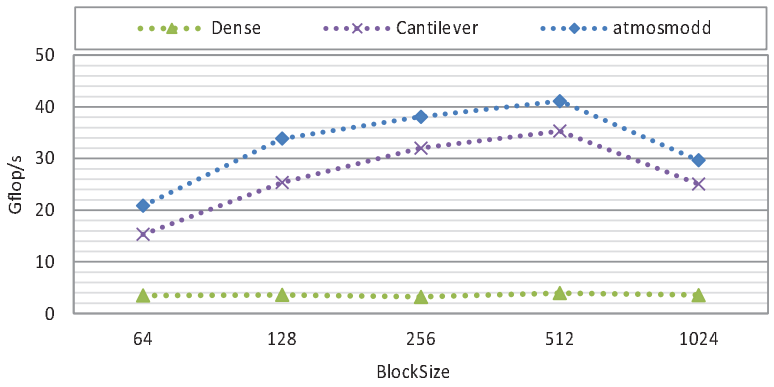
In this section, we first analyze the performance gain achieved for the basic formats under different architectural configurations. Table I shows the best architectural configuration for each format. All experiments throughout this study are conducted on GTX480 with 480 cores. It works with a clock frequency of 1.4GHz, and has 1.5 GB DRAM with 177GB/s bandwidth. Nvidia CUDA SDK 6.5 was used for all experiments.

TABLE A. I. THE BEST CONFIGURATION TO ACHIVED BEST PERFORMANCE FOR EACH FORMAT

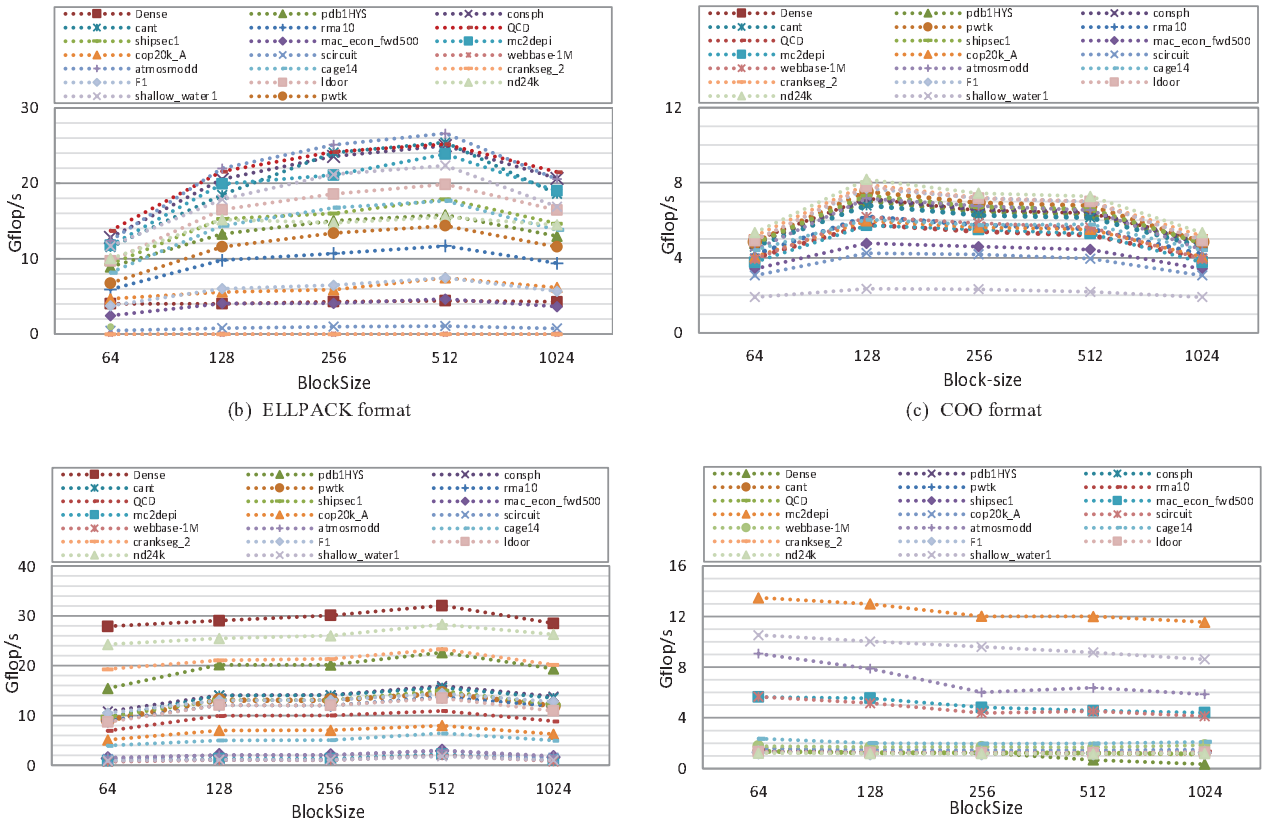
Formats	Block Size	Register (Single-Double)	Memory
DIA	512	20-33	L1
ELLPACK	512	20-33	L1
COO	128	20-33	L1
CSR(single)	64	20-33	L1
CSR(vector)	512	20-33	L1

A.1.1 Block Size

The impact of block-size on the performance is shown in Fig.A.1. In this figure, the best performance for ELL, DIA and CSR-vector formats is achieved with 512 threads inside a block. Considering the maximum number of blocks inside an SM which is 8 and the number of active threads inside an SM (i.e., 1536), the number of active threads will be 1536 (for compute capability 2.x) and occupancy degree equals 100%.



(a) DIA format



(b) ELLPACK format

(c) COO format

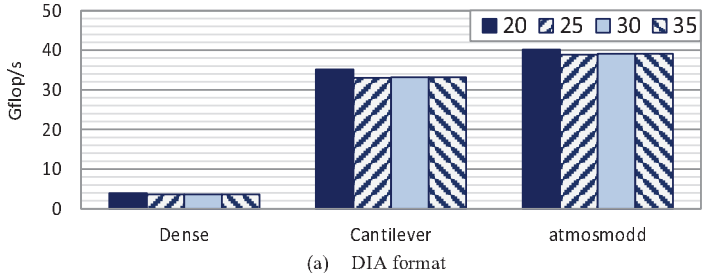
(d) CSR(vector) format

(e) CSR(scalar) format

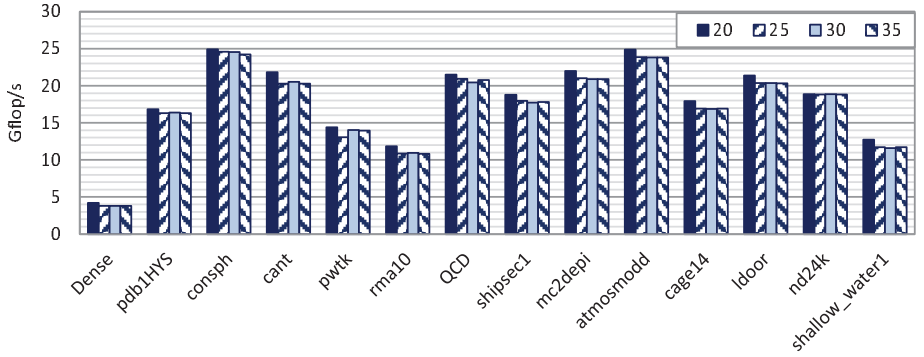
Fig A.1. The performance achieved by basic formats for different blocks sizes

A.1.2 Number of Registers

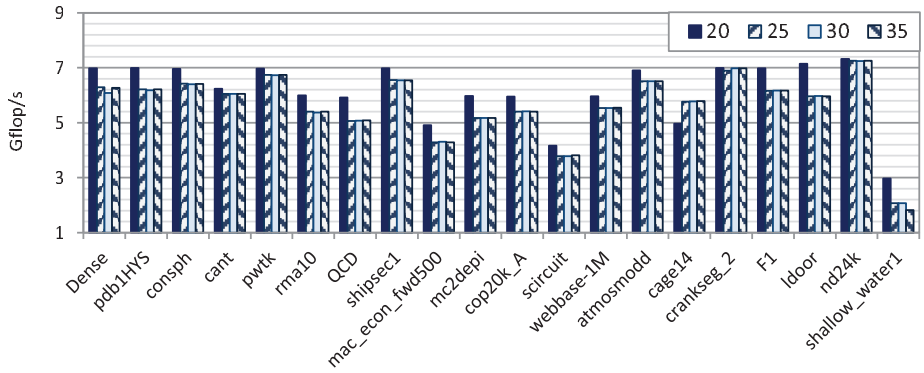
Effective register allocation has become a key to major improvements in program performance. Fig.A.2. shows the effect of the number of allocated registers on SpMV performance. As shown in the figure, as expected, by decreasing the number of allocated registers, a significant improvement in SpMV performance can be achieved without increasing the number of total spills. Table II shows the measure of occupancy by the limited number of registers of each SM (i.e., 32768 for compute capability 2.x). Regardless of block-size, the compiler allocates 27 and 33 registers to each thread for single and double-precision computation, respectively.



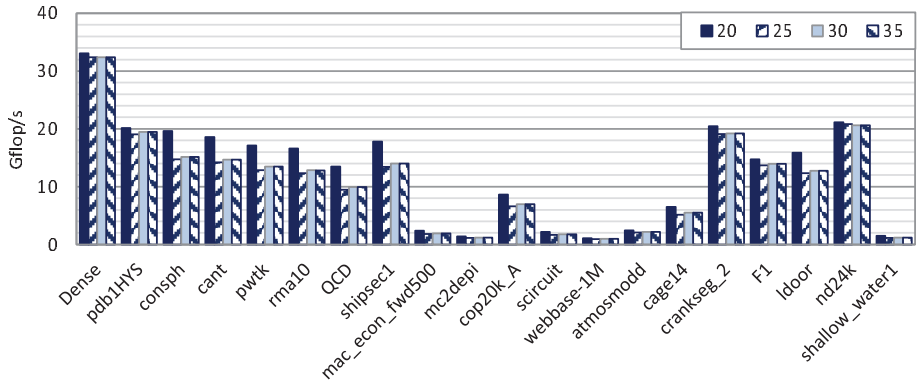
(a) DIA format



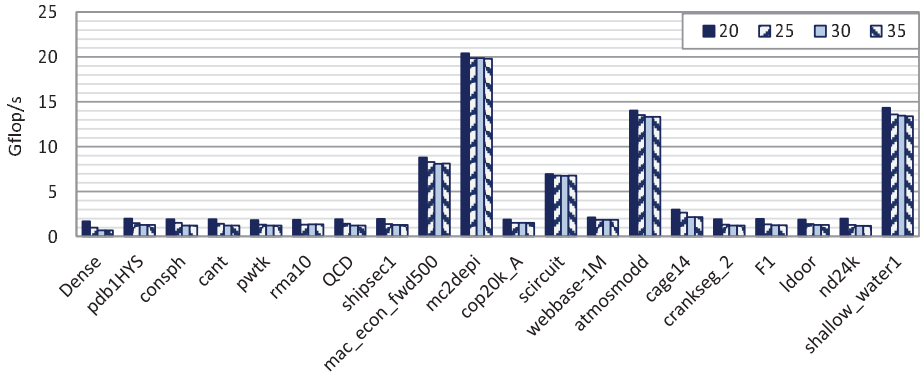
(b) ELLPACK format



(c) COO format



(d) CSR(vector) format



(e) CSR(scalar) format

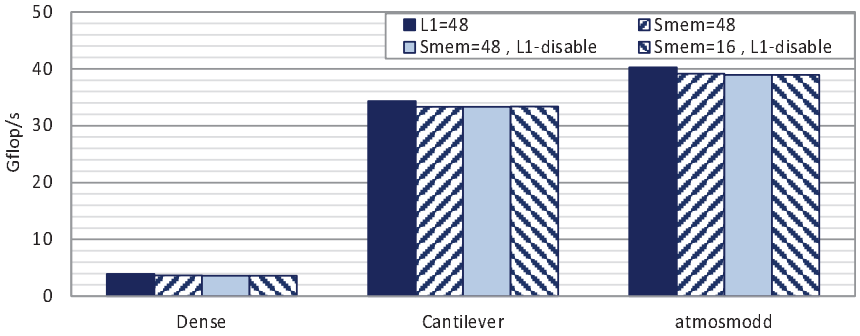
Fig A. 2. The effect of the number of registers on the performance for basic formats

TABLE I. OCCUPANCY RATE FOR DIFFERENT NUMBER OF ALLOCATED REGISTERS

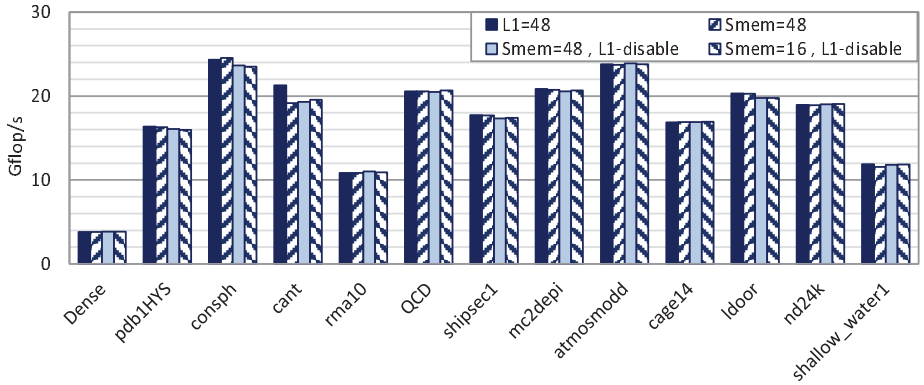
Number of Registers	20	25	30	35
Occupancy for Single Precision (%)	100	66	66	66
Occupancy for Double Precision (%)	100	66	66	33

A.1.3 Memory Hierarchy

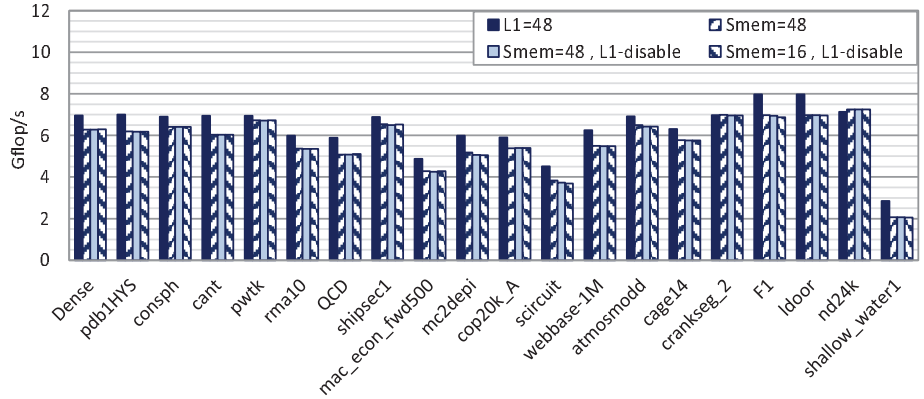
The impact of memory configuration on the performance is shown in Fig A.3 and Fig A.4. It is worth mentioning that Fermi architecture does not support the new setting of 32KB shared-memory/32KB L1-cache. As shown in the figures, the performance of CSR(vector) increases when the size of L1 cache is set to 48KB.



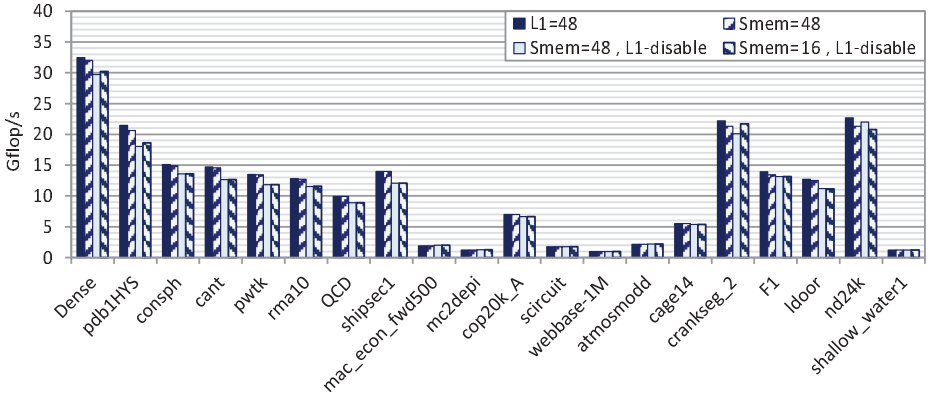
(a) DIA format



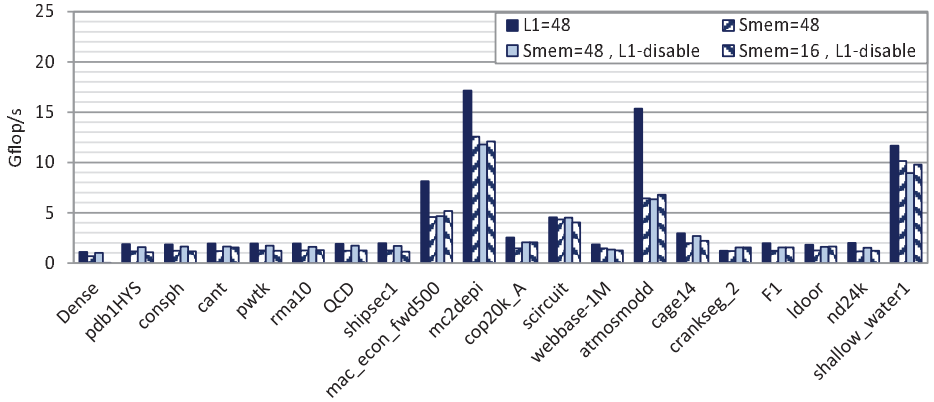
(b) ELLPACK format



(c) COO format

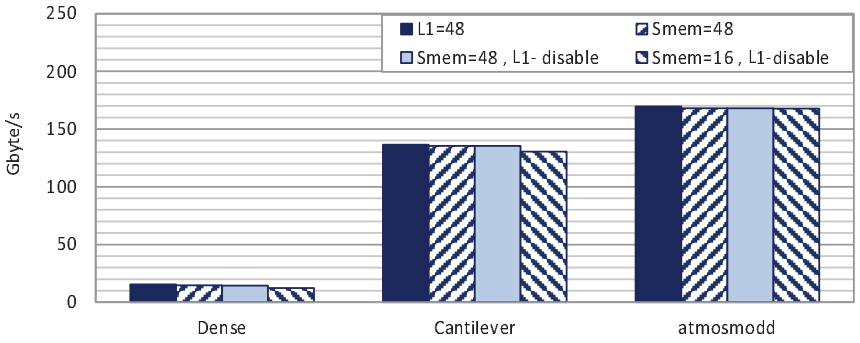


(d) CSR(vector) format

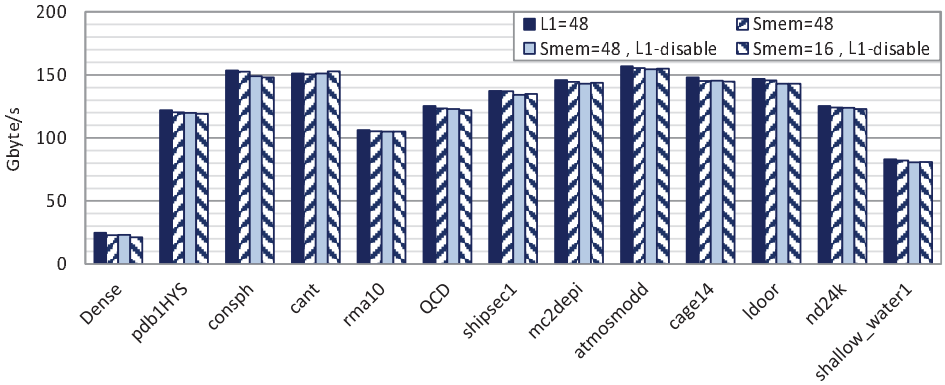


(e) CSR(scalar) format

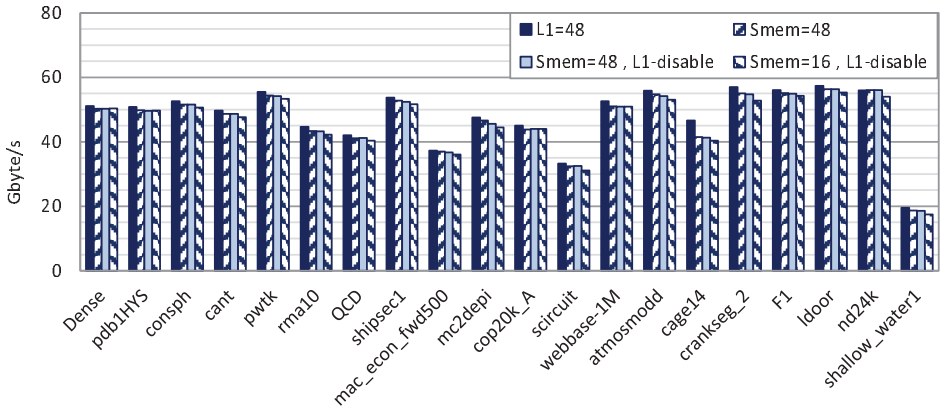
Fig A.3. The impact of different memory configuration on basic formats



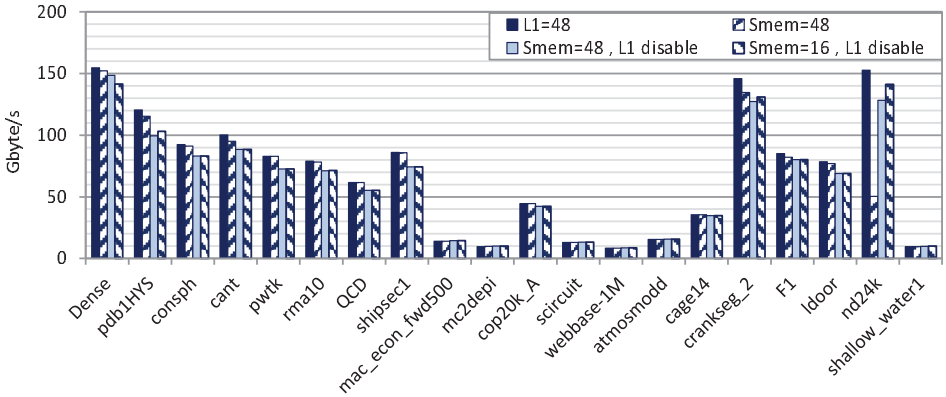
(a) DIA format



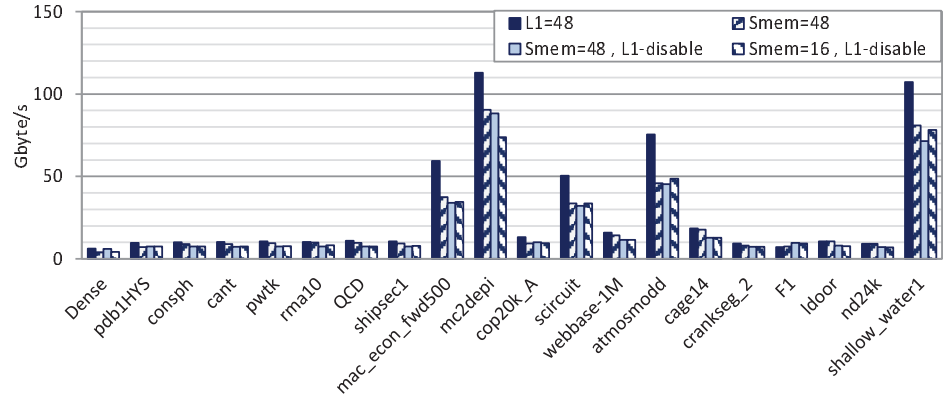
(b) ELLPACK format



(c)COO format



(e) CSR(vector)



(d) CSR(scalar)

Fig A. 4. Bandwidth results for matrices with different memory configuration on basic format

A.2 Adaptive Run-time System

In order to illustrate the practicality of our auto-tuner under Fermi architecture, Figures A.5 and A.6 show the performance of SpMV computation for various sparse matrices. The effective parameters were set based on previous experiments. In these figures, the accuracy of our system to choose a proper format is illustrated.

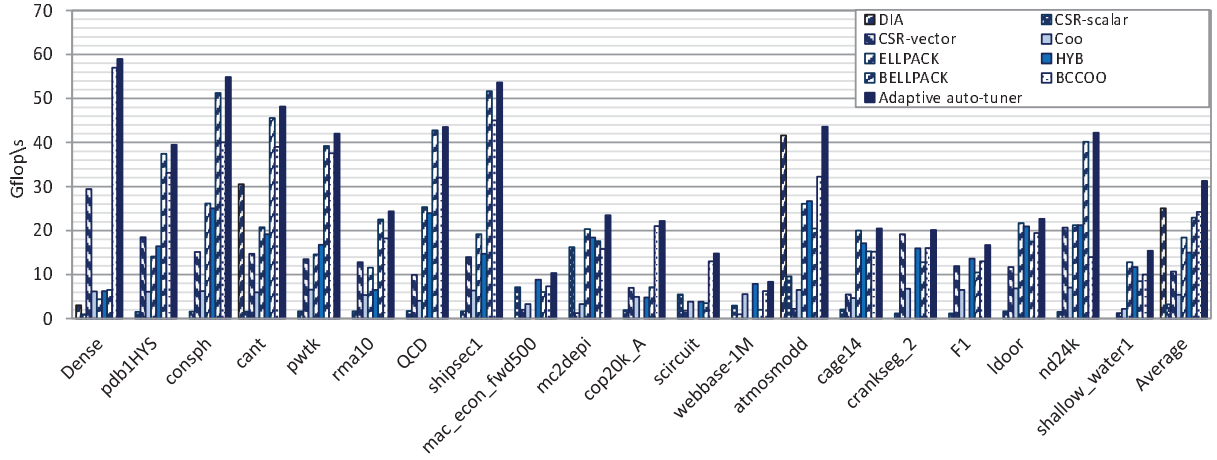


Fig A. 5. Computational power for the adaptive scheme

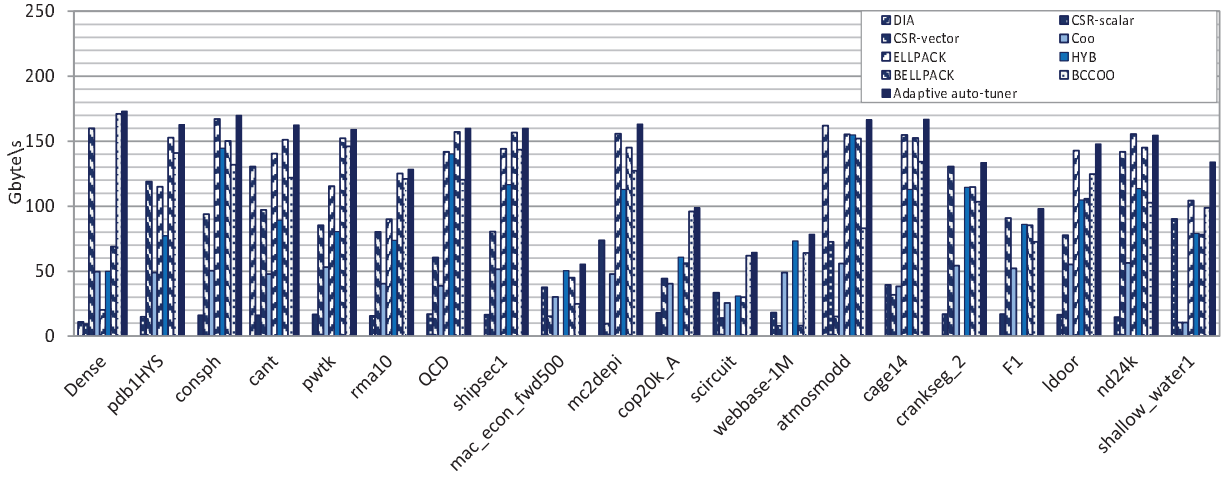


Fig A. 6. Effective bandwidth for different schemes

References

[1] NVIDA. Whitepaper NVIDIAs Next Generation CUDA Compute Architecture: Fermi, 2009.
[2] NVIDIA. Compute Visual Profiler User Guide, 2013.
[3] NVIDIA. NVIDIA CUDA C Programming Guide, 2013.
[4] NVIDIA Corporation. Tuning CUDA Applications for Fermi, Version 1.0, 2010.

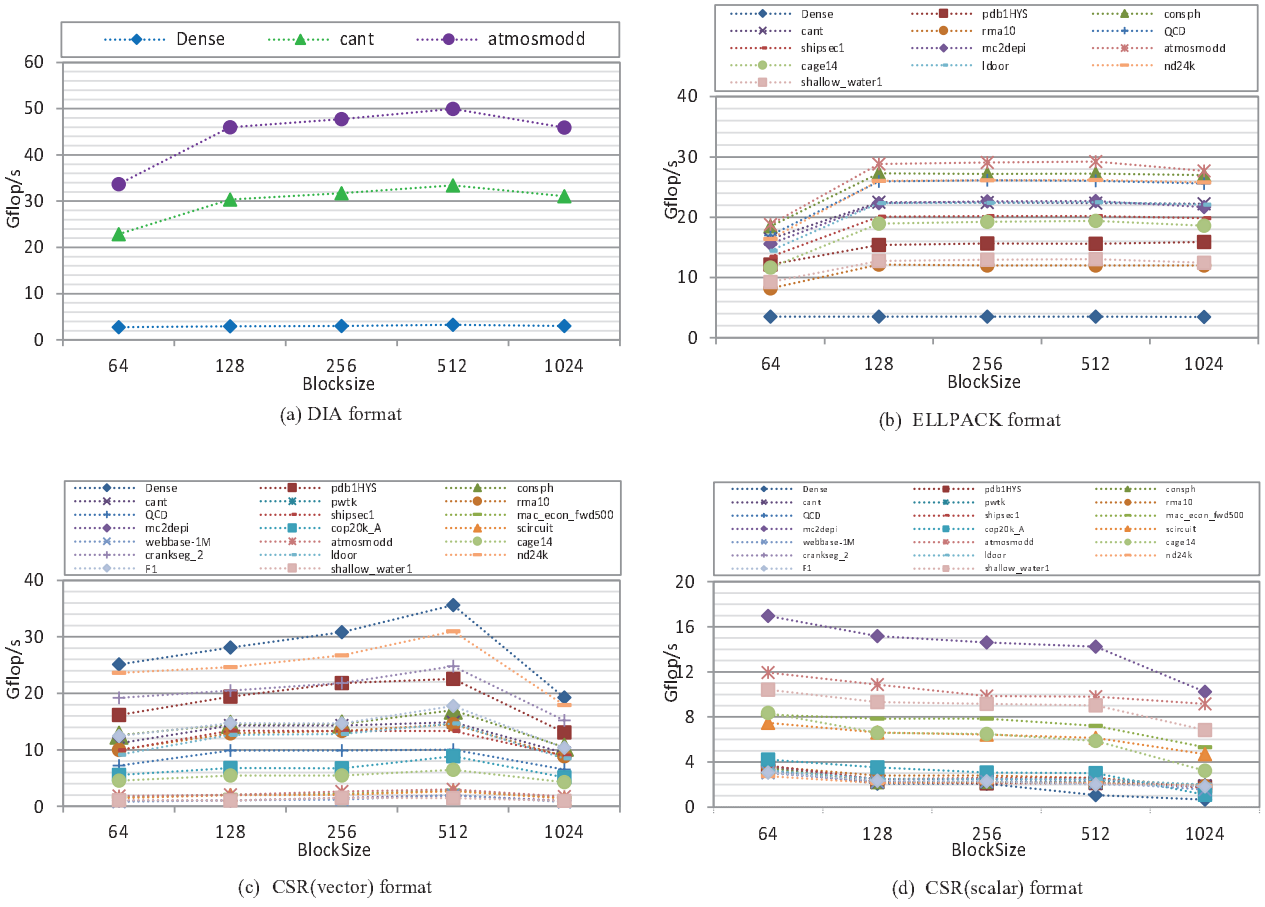
APPENDIX B. *ANALYSES UNDER KEPLER PLATFORM*

This appendix contains the results of using the proposed auto-tuner in Kepler platform.

B.1 Experiments to Set Parameters

B.1.1 Block Size

The impact of block-size on the performance is shown in Fig.B.1.



(a) DIA format

(b) ELLPACK format

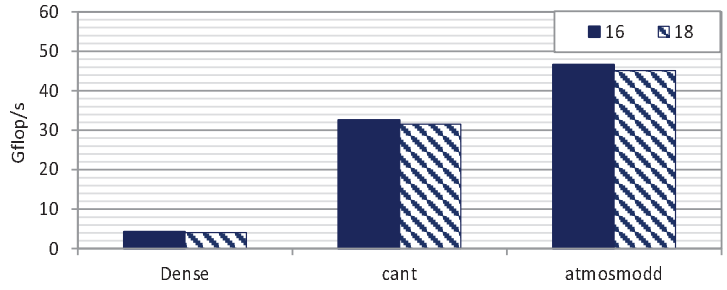
(c) CSR(vector) format

(d) CSR(scalar) format

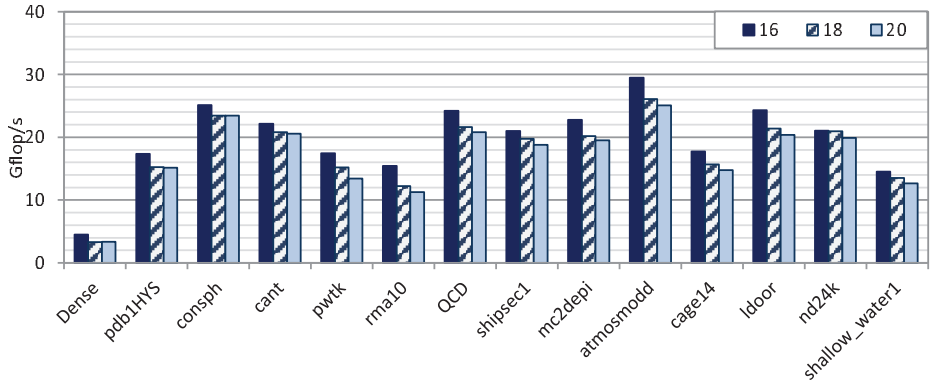
Fig B. 1. The performance achieved by basic formats for different blocks sizes

B.1.2 The Effect of the Number of Registers

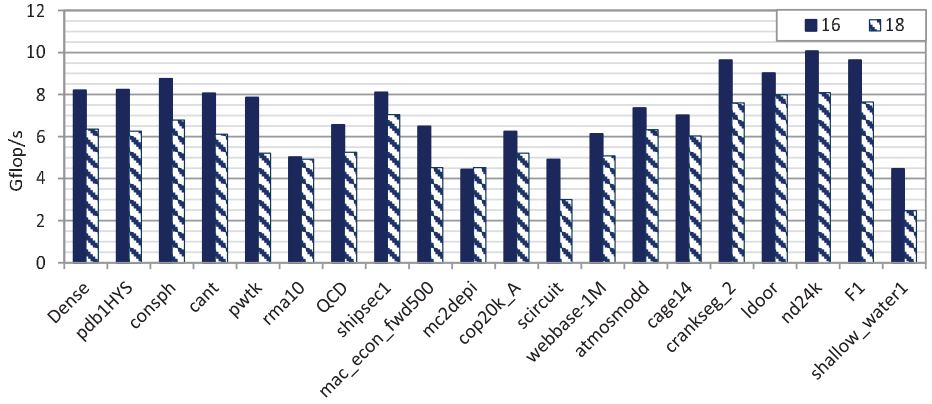
Fig.B.2. shows the effect of the number of allocated registers on SpMV performance.



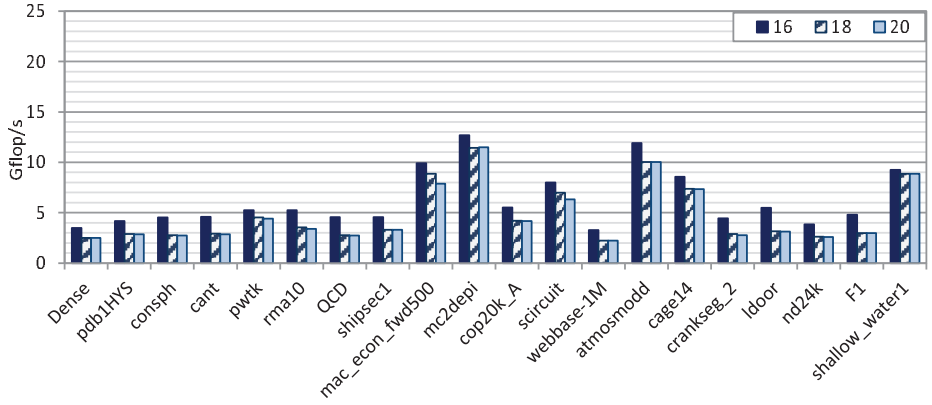
(a) DIA format



(b) ELLPACK format



(c) COO format

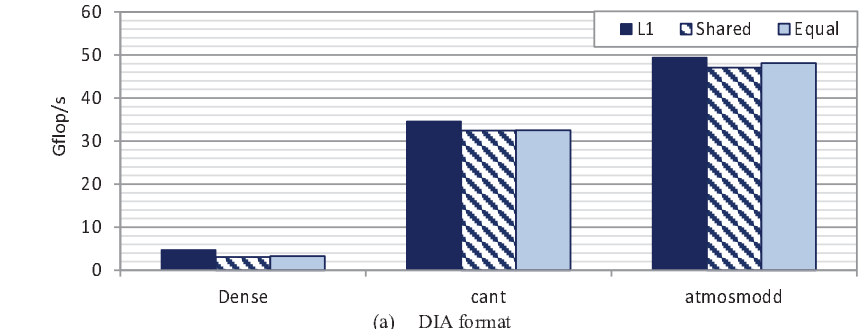


(d) CSR(scalar) format

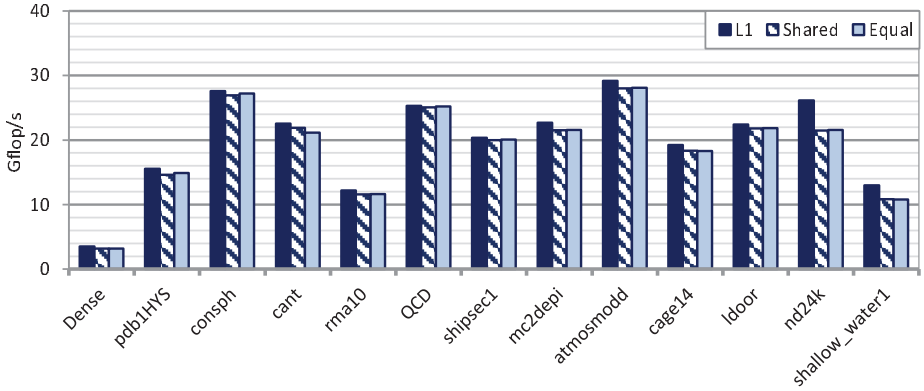
Fig B. 2. The effect of the number of registers on the performance for basic formats

B.1.3 Memory Hierarchy

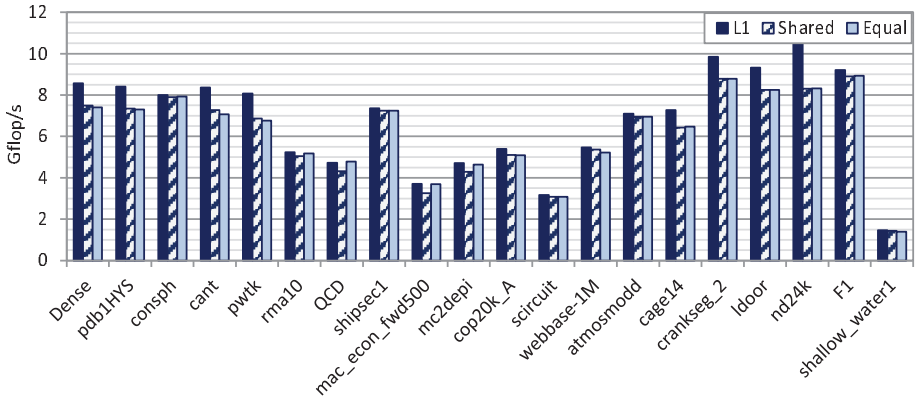
The impact of memory configuration on the performance is shown in Fig. B.3 and Fig. B.4.



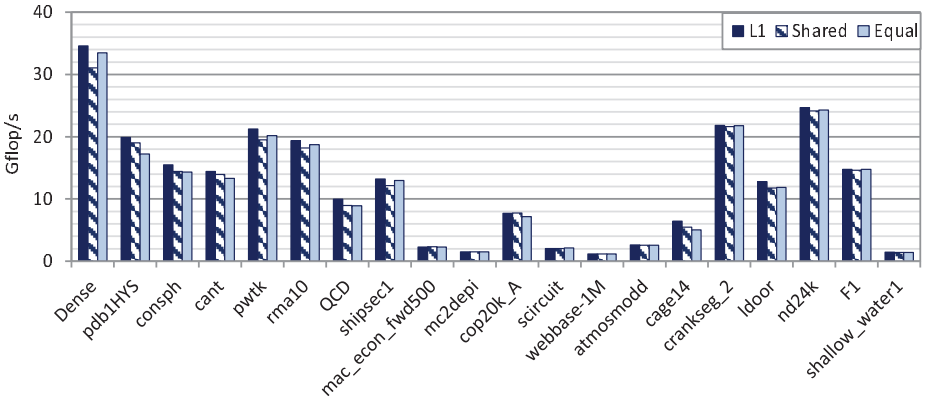
(a) DIA format



(b) ELLPACK format

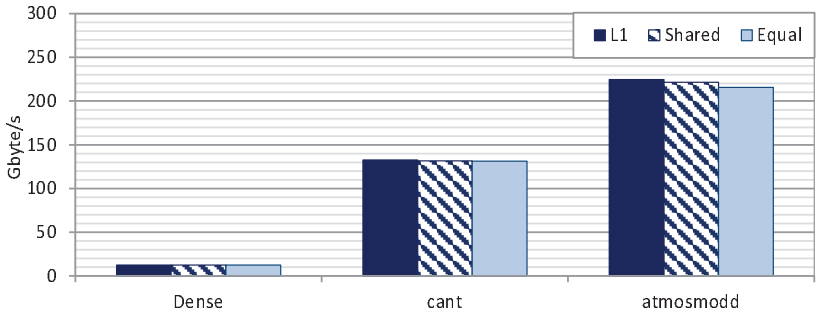


(c) COO format

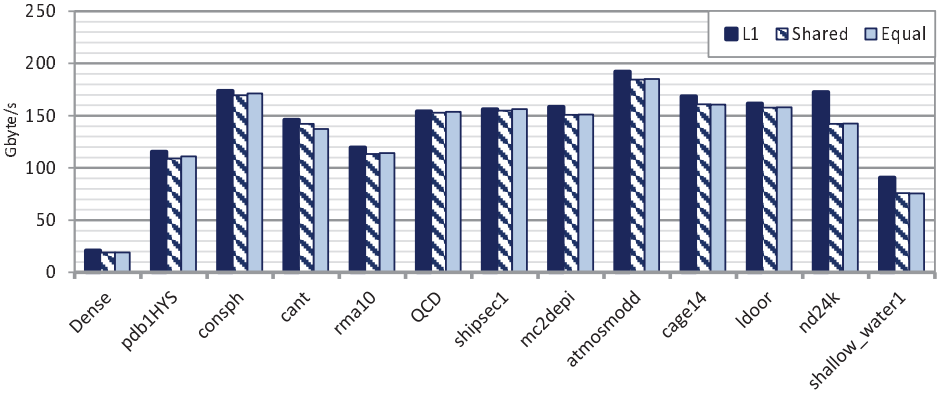


(d) CSR(Vector) format

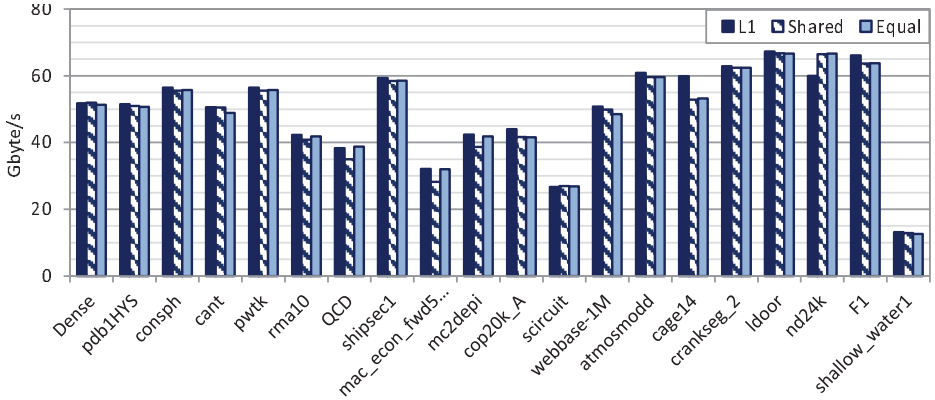
Fig B. 3. The impact of different memory configurations on the performance for different basic formats



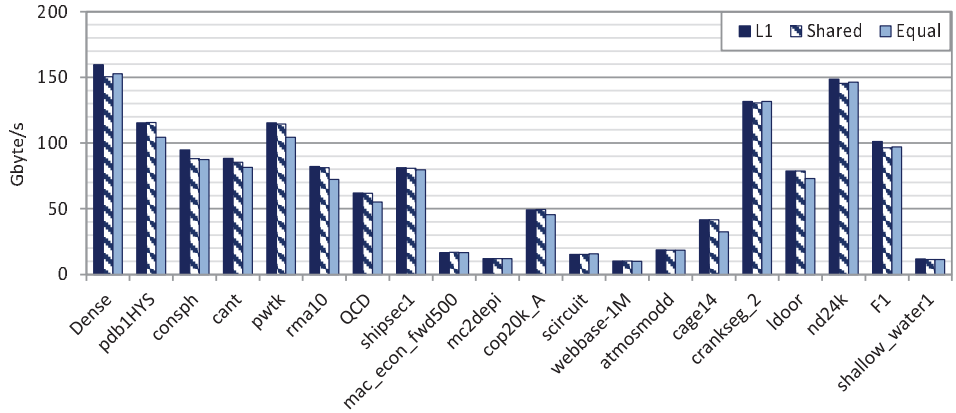
(a) DIA format



(b) ELLPack format



(c) COO format



(d) CSR(vector) format

Fig B. 4. . Bandwidth results for matrices with different memory configuration for different basic formats