Performance Evaluation of JFS Compression in a DB2 Environment

Ramesh Rajagopalan
Solutions Development,
IBM Web Servers, Beaverton, OR
rameshsr@us.ibm.com
October 2000

Abstract

In general, compression reduces the amount of disk space required to store a collection of data. We also expect it to save disk I/O, because if the data occupies less space we require fewer operations to access it. However, we need extra CPU cycles to compress the data before writing to a disk and to decompress the data after reading from a disk. We must balance the space and I/O gains against the CPU overhead in any given scenario. Data compression is widely used in the mainframe world. Experiences with IBM S/390, which supports hardware-enabled compression, have shown that it is beneficial to use compression. Hardware assisted compression significantly reduces the extra CPU cycles needed for the compression/decompression task. AIX Journaled File System (JFS) supports data compression. We wanted to evaluate the impact of compression in a UNIX environment. In this paper we discuss the results we obtained by using JFS compression in a database environment.

Introduction

We chose the industry standard TPC-H Benchmark to study the effect of JFS compression using DB2. The TPC-H Benchmark is a Business Intelligence (BI) benchmark. The benchmark represents an ad-hoc query environment. Unlike the On-Line Transaction Processing (OLTP) workloads which are characterized by random reads and updates of small blocks of data, the BI or Decision Support System workloads are characterized by sequential reads, big block reads on very large tables and relatively very few batch updates.

The TPC-H workload measures a system's capabilities through various database operations. For example, full table scans, sorting, hash joins and aggregation. In the BI environment, I/O throughput is crucial to achieve the query throughput and query response time. We wanted to understand how the JFS compression influences the I/O throughput. We expected:

- considerable disk space savings
- disk I/O gains (higher throughput or better latency)
- increased CPU activity due to decompression/compression

In the following sections, we provide the system configuration, explain the workload, describe the database layout, analyze the results and observations, and summarize the future work and the conclusions.

Configuration

Table 1 shows the configuration of the system used for the performance study.

Machine	IBM S70A
#CPUS	12 POWER PC II
Real Memory	32 GB
Disk Subsystem	1 SSA adapter with two loops, each having 16 4G SSA logical disks
Operating System	AIX 4.3.3
Database	DB2 V7.1 FP1

Table 1 - System Configuration

Workload

In order to conduct the test, we chose a small-scale 10 SF (~10 GB) TPC-H database. We ran the TPC-H power queries without the update functions that is **single stream and no UF**. In order to get consistent results, we executed the power test twice successively. We obtained the results by running the benchmark twice, by building the database on uncompressed file systems and on compressed file systems.

Database Layout

We created one tablespace with 16 containers to contain the TPC-H tables and indexes. This was a SMS tablespace named TPCH. Figure 1 shows the relationship among the different storage objects for the TPCH tablespace.

We specified the following values for creating JFS:

NBPI=16384, AG=8, FRAG=512

We disabled/enabled compression while creating JFS via the COMPRESS parameter.

For the TPCH tablespace, we specified

PAGESIZE 4K, EXTENTSIZE 32, PREFETCHSIZE 512

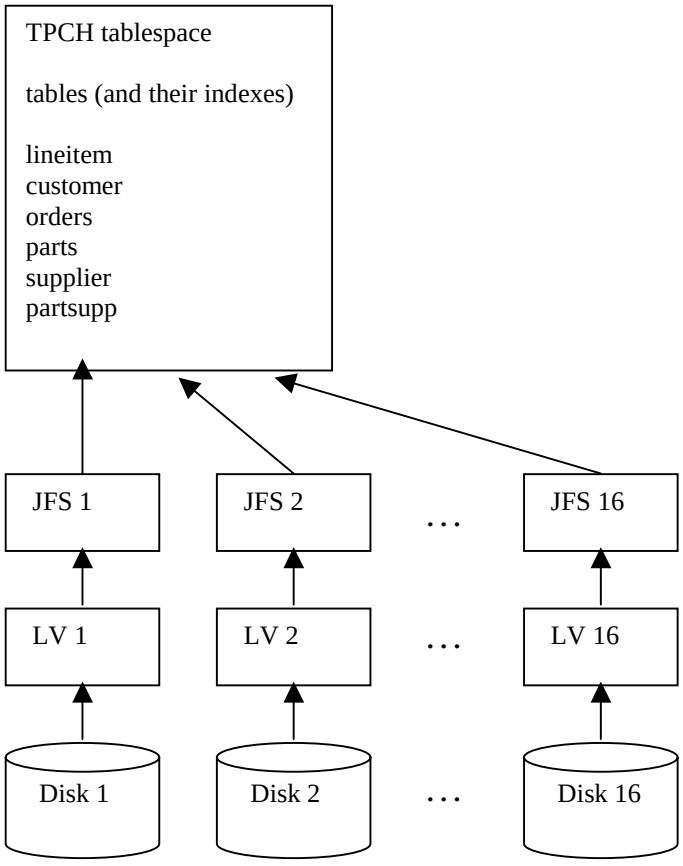

Figure 1 – TPCH tablespace storage objects

We tuned the database manager and database configuration settings that impact table scans and sorting. In particular, we set the database configuration NUM_IOSERVERS to 16. We had the DB2 registry setting

DB2_MMAP_READ set to YES. This facilitates transferring the data from the files directly to the database buffer pool bypassing the kernel buffer cache.

We allocated 4 disks each for logs and temporary tablespace. Figure 2 shows the relationship among the different storage objects for the logs and temporary tablespace. **In order to simplify the study, we did not enable compression for either logs or temporary tablespace.**

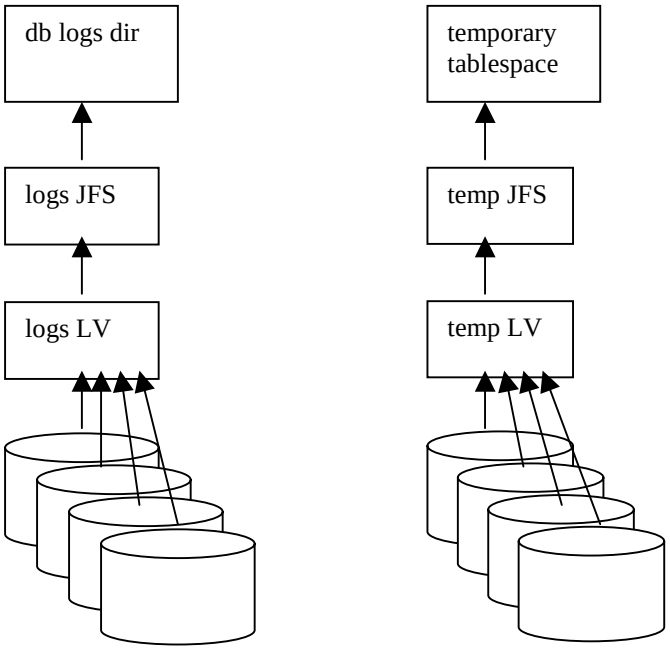

Figure 2 – logs and temporary tablespace storage objects

We used the remaining 8 disks for the UFTEMP and TSNODE_1 tablespaces.

Results and Observations

Compressibility of data

Figure 3 shows the database disk usage with compression.

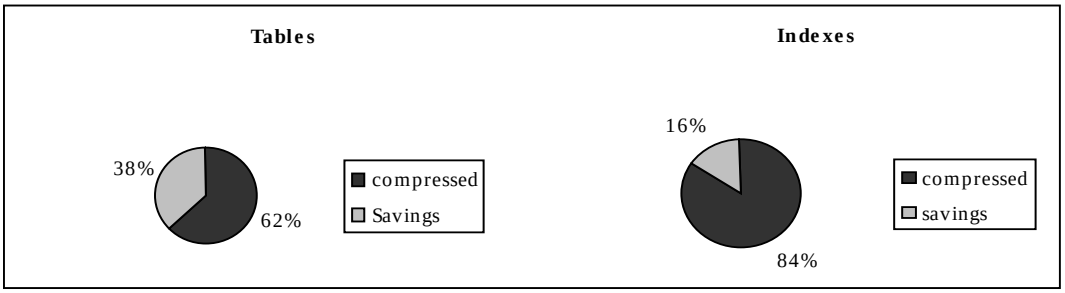

Figure 3 – Database disk usage with compression

We see that the compression reduces the database size to 67%, saving 33% disk space. This is the significant saving in disk space we expected to obtain. We provide the size of the database without and with compression in Table 2.

Size	w/o Compression	With Compression
Tables (GB)	11.42	7.10
Indexes (GB)	3.26	2.74
Total (GB)	14.68	9.84

Table 2 – Database Size

We noticed the following:

- with a fragment size of 2048, the saving in disk space is about 2%
- with a fragment size of 1024, the saving is about 20%
- with a fragment size of 512, the savings is about 33%

In order to have the maximum disk space saving we chose the smallest fragment size of 512.

Query Performance

We provide the key TPC-H metrics in Figure 4.

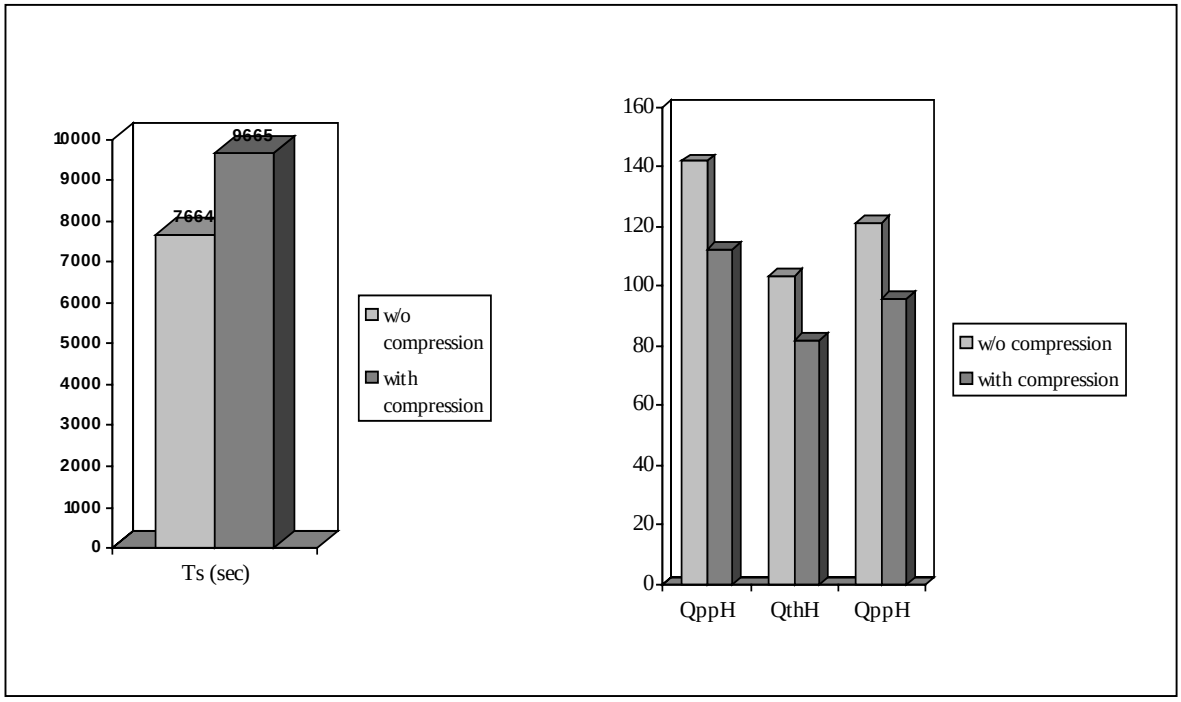

Figure 4 – TPC-H metrics

Based on the results from the power runs without UF, we observed that the performance degrades with compression. The performance with compression is 21% slower compared to the performance without compression. We provide the TPC-H metrics and query times without and with compression in Table 3 and Table 4.

TPC-H Metrics	w/o Compression	With Compression
Ts (sec)	7664	9665
QppH@10.0GB	141.9	112.2
QthH@10.0GB	103.3	82.0
QppH@10.0GB	121.1	95.9

Table 3 – TPC-H Metrics

Query	Q14	Q2	Q9	Q20	Q6	Q17	Q18	Q8
w/o compression	238.6	40.4	639.4	92.8	423.3	73.6	480.3	570.2
with compression	262.4	52.0	779.3	141.4	602.4	81.6	567.6	609.4
Query	**Q21**	**Q13**	**Q3**	**Q22**	**Q16**	**Q4**	**Q11**	**Q15**
w/o compression	720.0	317.6	386.3	19.3	37.0	416.9	452.2	358.8
with compression	970.0	336.1	521.0	29.1	46.2	572.7	475.1	434.4
Query	**Q1**	**Q10**	**Q19**	**Q5**	**Q7**	**Q12**		
w/o compression	350.9	428.7	306.6	445.3	381.3	484.0		
with compression	433.3	550.3	390.3	605.4	521.2	682.4		

Table 4 – Query Times

We needed to tune the VM *maxrdahead* parameter to obtain better disk I/O throughput. We obtained a maximum throughput of 28 MB/sec (without compression) for the queries that were doing full table scans.

We noticed that the maximum disk I/O throughput was about 14 MB/sec with compression. This is about 50% compared to that obtained without compression (maximum disk I/O throughput obtained without compression is 28 MB/sec). However, we expected it to be at least 65%, as the data is compressed to 65%. We need to investigate this further in order to explain this phenomenon.

We observed that, with compression the system time (%sys column in the output of sar or sy column in output of vmstat) for most of the queries was about 1.5x compared to that without compression. This is the overhead we expected to incur due to the decompression task that is taking extra CPU cycles. The decompression task runs in the kernel space, thus showing up as increased system time.

Future Work

We did not attempt to characterize the performance impact due to compression with update queries. It would be interesting to perform this study, as in the real world update transactions would be executed against the database. It would also be interesting to observe the effect of compression on database log files with update queries.

Conclusion

We demonstrated that using JFS compression in a DB2 environment with the TPC-H workload yields a disk savings on the order of 33% and impacts the performance on the order of 20%. We could infer that if moderate performance degradation could be tolerated, using JFS data compression would be beneficial with real world databases, against which read-intensive queries are executed, provided the compressibility of data is significant.

Acknowledgements

This work could not have been done without help and support from many individuals. Salvatore Vella of DB2 UDB development was instrumental in the initiation of the work described in the paper. Enzo Cialini of DB2 UDB system test provided the machine. He also provided valuable suggestions and feedback that were incorporated in this paper. My manager Laurent Montaron of Solutions development [DB2 database engineering] was very supportive of this work. I had many useful technical discussions with him. He also provided me valuable suggestions and feedback that were incorporated in this paper. Finally, many members of DB2 UDB development, performance, benchmarking, system test and development support team at Beaverton and Toronto provided valuable advice and support in understanding SSA disk configuration on the test machine, TPC-H and DB2 UDB. They also helped me in numerous ways in various phases of the work.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States or other countries or both:

AIX
JFS
SSA
POWER
S70A
DB2
DB2 Universal Database
IBM

TPC-H, QppH, QthH, and QphH are trademarks of the Transaction Processing Performance Council.

Other company, product, and service names may be trademarks or service marks of others.

