

Performance Study of Journalling Filesystems

Nicolas S. Wettstein
nicolasw@student.ethz.ch

WS 01/02

Course project
Department of Computer Science
ETH Zürich

Lecture: Computer Systems Performance Analysis and Benchmarking
Lecturer: Prof. Thomas M. Stricker
Teaching Assistant: Christian Kurmann

Abstract

The last years have witnessed the upcoming of a new type of filesystems offering faster recovery than their conventional counterparts (FAT, ext2fs, etc.). The price to pay for faster recovery is the additional journalling overhead required by every operation. The performance study presented in this report tries to measure the effect of this overhead on large file-I/O. The two journalling filesystems ext3fs and ReiserFS are compared to two different versions of ext2fs. The report contains an overview of the project, a statistical summary and interpretation of the measurements as well as an experimental design revealing the quality of the evaluation.

Contents

1	Introduction	3
1.1	Why journalling?	3
1.2	Goal	3
1.3	Limitations	3
2	System Setup	4
2.1	Component Under Study	4
2.1.1	Services	4
2.2	Preformance Metrics	4
2.3	Factors	4
2.3.1	Filesystem	5
2.3.2	Access method	5
2.4	Workload and evaluation technique	5
3	Results	5
3.1	Bonnie benchmark	5
4	Conclusion	7
4.1	Lessons learned	7
A	Bonnie benchmark	9
A.1	Sample data	9
A.2	Scatter plots of sample data	10
A.3	Statistical summary of sample data	10
A.4	Normal quantile-quantile plots for sample data	11
A.5	Two-factor full factorial design with repetitions	12

List of Figures

1	The sample means with 99% confidence intervals	6
---	--	---

List of Tables

1	Configuration of the test machine.	3
---	--	---

1 Introduction

This section explains the concept of journalling filesystems and states the goal, the assumptions, and the limitations of this study.

1.1 Why journalling?

Today's operating systems use buffer-caches to accelerate access to the data stored on harddisks. Consequently, modifications have not been made to buffered data and are not instantly made persistent (i.e. written back to the disk). This can become a problem if the computer crashes when part of the modifications are not written back yet. Besides data loss, there is also the possibility of corrupting the filesystem when the filesystem's data and metadata do not correspond. To find and repair such inconsistencies the recovery program needs to scan the entire disk, a very time-consuming task. Considering the exponential growth of disk space, it is even bound to become infeasible in practice. Journalling filesystems achieve faster recovery by using the concepts of *ordered transactions* and *logging*, as often used in database systems [SFS97, Chapter 15]. The idea is to ensure that all filesystem operations are atomic and to record them in a non-volatile structure called the *journal*. The journal has only to be big enough to track the operations that have not been committed, i.e. made persistent yet (the default journal size for ext3 is 10MB). Recovery now consists only of checking the subset of data and metadata that has been modified recently, i.e. is recorded in the journal. The duration of recovery is not any more a function of disk size but a function of the number of uncommitted operations possible in the system, which is much lower.

1.2 Goal

This study evaluates the I/O performance of journalling filesystems (ext3fs and ReiserFS) with respect to non-journalling filesystems (two different versions of ext2fs). Table 1 lists the configuration of the computer used for benchmarking.

<i>Model</i>	Dell Precision 410
<i>Operating system</i>	SuSE Linux 7.1
<i>CPUs</i>	2 x Pentium II 400 MHz
<i>Memory size</i>	256 MB
<i>SCSI adapter</i>	Adaptec AIC-7890/1 Ultra2 SCSI host adapter
<i>Harddisk model</i>	Seagate Cheetah ST-39102LW Ultra2 SCSI Wide

Table 1: Configuration of the test machine.

1.3 Limitations

This performance project is a practical exercise for a lecture¹. Therefore its depth is very limited and certain efforts have been made mostly for the sake of applying theoretical concepts discussed in lecture and in [Jai91]. Also, only

¹The lecture is called *Computer Systems Performance Analysis and Benchmarking* and it is held by Prof. Stricker at the ETH Zürich.

a subset of the computed statistics has been analyzed. Large file-I/O is only one of many things we expect a filesystem to do well. The results presented in this report should be interpreted in this angle. It is recommended to additionally consult other performance evaluations of journalling filesystems so as to complete the picture.

2 System Setup

In the attempt of a systematical approach, this section lists the major characteristics of the performance study according to [Jai91, Section 2.2]. It is a more detailed version of the project plan established prior to evaluation.

2.1 Component Under Study

For the purpose of this evaluation, the *Component Under Study* (CUS) is defined as the subset of the operating system that provides block-based I/O and filesystem management.

2.1.1 Services

A filesystem provides 3 basic services to the user:

1. file read access
2. file write access
3. directory services (e.g. directory creation)

Due to time limitations, directory services will not be considered. For all 3 service requests, the following outcomes are possible:

1. done correctly
2. done incorrectly
3. cannot do (e.g. disk full when trying to write)

Due to time limitations, this study considers only correct operation.

2.2 Performance Metrics

The criteria chosen for comparing the different filesystems is:

Productivity Measuring *throughput*, i.e. the number of bytes read or written during a fixed period of time, will provide information about the filesystem's productivity. This is a *higher is better* metric and will be measured in MB/s.

2.3 Factors

The following factors will be varied:

2.3.1 Filesystem

This is the primary factor of the study. There are two representatives of each filesystem type, journalling and non-journalling.

Ext2fs When the *Second Extended Filesystem* was first released in 1993 it was intended to fix several problems of its predecessor, the *Extended File System*. The main problems were fragmentation and bad performance due to the use of linked lists for managing free blocks and inodes. The basic concepts of ext2fs are drawn from the Unix operating system [Bac86]. It uses a block-oriented allocation scheme. Ext2fs has become the standard Linux filesystem. For more detailed information, see [CTT95].

Two different Linux kernels (2.3.0 and 2.4.17) have been compiled so that two versions of ext2fs can be used to represent non-journalling filesystems.

Ext3fs adds journalling capabilities to ext2fs while remaining entirely compatible to it. The filesystem journal is kept in a special file (using a reserved inode) and it keeps records of all operations defined in the Virtual File System (VFS) as well as of data updates. For more detailed information, see [Twe98].

The version of ext3fs included in the 2.4.17 kernel is first representative of journalling filesystems.

ReiserFS uses balanced trees to organize files. This approach is claimed by the authors to be faster and more space-efficient than conventional block-based filesystems like the ones mentioned above. For more details, see [Rei01].

The version of ReiserFS included in the 2.4.17 kernel is the second representative of journalling filesystems.

2.3.2 Access method

It was difficult to find a second factor that is parametrized by the available benchmarks and would reveal significant differences between the filesystems. In the hope that read access would prove more expensive on journalling filesystems, *read-only* and *write-only* disk accesses are distinguished.

2.4 Workload and evaluation technique

Bonnie Bonnie tests the throughput and the CPU usage of sequential read, sequential write, and random seek operations on a big file. The read and write operations are done with both character-based and block-based I/O.

The Bonnie benchmark is used for measuring the *throughput* of the filesystems. The results of block-based read and write operations are evaluated using a *two-factor full factorial design with replications* [Jai91, Chapter 21].

3 Results

3.1 Bonnie benchmark

The Bonnie benchmark was run with 10 repetitions for each filesystem. File size was set to 1024 MB. After each repetition, the filesystem has been unmounted,

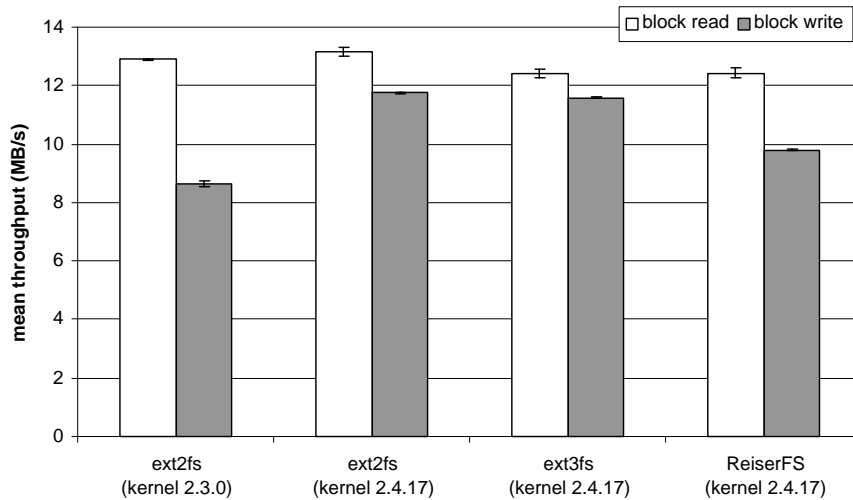


Figure 1: The sample means with 99% confidence intervals

recreated, and remounted in order to reproduce the exact same condition for each run.

The sample data is listed in Appendix A.1². According to the scatter plots of the sample data, there are no outliers that should be ignored. Although some of the quantile-quantile plots for the sample data (see A.4) show slight asymmetries (plots (d) and (e)) and short tails (plots (g) and (h)), it still seems appropriate to assume that the samples are normally distributed (plots (a), (b), (c), and (f) fit well). Accordingly, the arithmetic mean has been chosen as index of central tendency and the standard deviation as index of dispersion (see A.3).

A look at the means and their respective 99% confidence intervals (see Figure 1) reveals the following (all of the following statements are made with at least 99% confidence):

- The write performance of ext2fs has improved by about 3 MB/s from version 2.3.0 to 2.4.17.
- The non-journaling filesystems outperform their counterparts by about 0.5 MB/s at reading.
- ext3fs and ext2fs (of the same version) are equally performant at writing, but ext2fs is about 0.6 MB/s faster at reading. This observation is rather counter-intuitive. I would have expected the contrary, both being equally fast at reading but ext3fs being slower at writing, since it has the additional overhead of maintaining the journal and otherwise uses the same code as ext2fs.
- Both journaling filesystems are equally fast at reading, but ReiserFS is about 1.5 MB/s slower at writing. This result was expectable, since the

²The Bonnie benchmark returns the throughput values in KB/s. Throughout this report they have been scaled to MB/s.

major advantages of ReiserFS lie in the management of many small files whereas the Bonnie benchmark measures I/O on one very big file.

Experimental design

The effects of the two factors, filesystem and access mode, have been estimated using a *two-factor full factorial design with replications* (see Appendix A.5 according to [Jai91, Chapter 22]. Because the response range is small ($max/min \approx 1$) for all filesystems, it is appropriate to assume a linear model. The following conclusions can be drawn:

- An average filesystem has a read throughput of 12.7 MB/s and a write throughput of 10.4 MB/s (see A.5 (a)).
- The model explains about 82% variation (A.5 (c)). It is therefore usable, but the unexplained 18% should be investigated further.
- The access modes explain significantly more variation than the filesystem types. This indicates that variation of access modes is not a good secondary factor for the study and further investigation should be directed towards finding better secondary factors.
- The interactions explain less than 2% variation (A.5 (c)) and may be ignored.

Due to time limitations, the final checks of the model assumptions have been omitted.

4 Conclusion

- Further measurements should be undertaken to obtain more detailed results. Especially the management of multiple files as well as the directory services should be further investigated.
- If the evaluation of I/O performance is to be continued, emphasis should be placed on finding better secondary factors than access mode.
- According to the obtained results the overhead of journaling filesystems seems to be acceptable in situations when very long filesystem recovery times are experienced.
- Considering the interchangeability of ext2fs and ext3fs, it seems possible to easily upgrade some ext2fs partitions of a running system and gather practical experience. If ext3fs proves to slow, one can return to using ext2fs almost instantly.

4.1 Lessons learned

During this project, I have realized how time-consuming a performance study can be if it is to produce meaningful results. Both width and depth of such a project tend to grow as one gets more acquainted with the subject and it is important to keep them within certain boundaries. It is crucial not to become too entangled in the statistical details but rather try to see the big picture.

Acknowledgments

I would like to thank Prof. Stricker for an interesting lecture, Christian Kurmann for his help, Anders Fornander for cross-reading, and the students working in IFW E33 for their patience (the hard-drive of my test machine located in this room makes a lot of noise, especially when running filesystem benchmarks for hours)!

References

- [Bac86] Maurice J. Bach. *The Design of the UNIX Operating System*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 1986.
- [CTT95] Rémy Card, Theodore Ts'o, and Stephen Tweedie. Design and implementation of the second extended filesystem. In Frank B. Brokken et al., editor, *Proceedings of the First Dutch International Symposium on Linux*. State University of Groningen, 1995.
- [Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, Inc., New York, NY, 1991.
- [Rei01] Hans Reiser. ReiserFS, 2001. <http://www.namesys.com>.
- [SFS97] Abraham Silberschatz, Henry K. Ford, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, 3rd edition, 1997.
- [Twe98] Stephen Tweedie. Journaling the Linux ext2fs filesystem. In *Linux-Expo '98*, 1998.

A Bonnie benchmark

A.1 Sample data

ext2fs (kernel 2.3.0)

run #	MB	sequential write						sequential read				random	
		per char		block		rewrite		per char		block		seeks	
		KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	/s	%CPU
1	1024	5698	99.3	8828	9.8	4683	11.6	5584	90.3	13292	14.0	146.9	3.1
2	1024	5686	99.0	8854	9.7	4714	11.5	5678	91.9	13336	14.0	149.3	3.4
3	1024	5687	99.0	8891	9.7	4742	11.7	5688	92.0	13454	14.2	149.4	3.4
4	1024	5628	98.1	8806	9.6	4736	11.4	5739	92.8	13161	15.1	148.6	3.5
5	1024	5709	99.5	8866	9.5	4750	11.7	5667	91.6	13076	15.6	149.9	3.3
6	1024	5618	97.8	8839	9.6	4718	11.5	5653	93.3	13093	15.5	146.9	3.7
7	1024	5707	99.3	8869	9.6	4736	11.3	5597	92.5	13148	16.1	151.2	3.3
8	1024	5699	99.2	8862	9.6	4698	11.5	5621	92.8	13138	15.8	147.3	3.1
9	1024	5700	99.3	8907	9.7	4690	11.2	5564	92.0	13018	15.5	148.5	3.4
10	1024	5707	99.3	8853	9.4	4688	11.3	5568	91.9	13308	16.3	150.9	3.1

ext2fs (kernel 2.4.17)

run #	MB	sequential write						sequential read				random	
		per char		block		rewrite		per char		block		seeks	
		KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	/s	%CPU
1	1024	4658	81.5	12202	11.7	6580	12.4	5977	96.5	13441	14.7	189.1	5.3
2	1024	4820	84.1	12178	11.7	6564	11.6	5952	96.2	13480	16.3	192.8	5.0
3	1024	4808	84.1	12210	11.7	6563	13.6	5938	96.0	13503	13.3	190.5	4.0
4	1024	4705	82.0	12059	11.5	6545	11.7	5886	97.0	13386	15.5	184.1	3.8
5	1024	4763	83.0	11980	11.3	6361	12.8	5902	97.4	13473	15.2	191.0	4.1
6	1024	4786	83.5	12161	11.7	6623	12.0	5838	96.2	13483	15.8	185.9	4.4
7	1024	4823	84.1	11998	11.6	6659	13.4	5998	97.1	13438	14.8	192.4	5.1
8	1024	4712	82.1	11661	11.2	6751	12.3	5941	97.9	13431	15.1	191.3	4.0
9	1024	4828	84.1	11848	11.3	6697	12.8	5928	97.9	13449	15.5	193.5	4.3
10	1024	4842	84.5	12092	11.5	6495	11.6	5861	96.5	13486	16.5	195.3	3.9

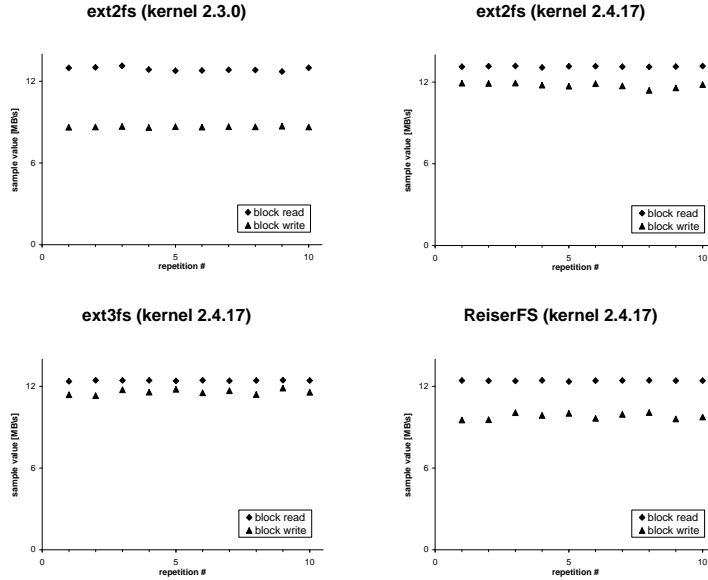
ext3fs (kernel 2.4.17)

run #	MB	sequential write						sequential read				random	
		per char		block		rewrite		per char		block		seeks	
		KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	/s	%CPU
1	1024	5231	98.2	11648	23.0	6395	13.2	5924	95.7	12667	12.5	170.5	4.0
2	1024	5156	96.6	11586	23.0	6437	12.7	6000	97.0	12743	14.0	173.0	4.6
3	1024	5227	98.5	12028	24.3	6467	13.8	5895	95.2	12736	14.3	170.3	4.2
4	1024	5256	98.7	11850	23.5	6531	13.8	5907	96.6	12736	13.3	168.9	3.5
5	1024	5164	96.9	12062	24.2	6442	13.0	5770	95.2	12694	15.7	170.2	3.4
6	1024	5154	96.8	11801	23.5	6311	12.6	5846	96.4	12744	17.9	170.5	3.2
7	1024	5196	97.6	11956	24.0	6422	13.2	5812	95.9	12706	15.3	171.5	3.7
8	1024	5266	98.8	11671	23.2	6467	13.7	5905	97.5	12727	15.1	169.9	3.4
9	1024	5212	98.1	12158	24.2	6426	12.6	5907	95.6	12753	15.6	169.0	3.3
10	1024	5206	97.7	11838	23.7	6483	13.4	5863	96.7	12728	15.3	171.7	3.9

ReiserFS (kernel 2.4.17)

run #	MB	sequential write						sequential read				random	
		per char		block		rewrite		per char		block		seeks	
		KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	KB/s	%CPU	/s	%CPU
1	1024	3820	73.9	9753	24.5	6426	13.3	6016	97.8	12722	14.5	180.1	5.1
2	1024	3736	72.3	9774	24.4	6694	12.4	6039	98.2	12708	16.5	181.1	5.3
3	1024	3811	73.7	10301	25.5	6425	12.7	5866	95.4	12698	14.4	180.8	4.5
4	1024	3816	73.8	10097	25.0	6389	12.4	5705	94.8	12735	16.4	181.1	4.5
5	1024	3798	73.3	10254	25.5	6516	11.8	5769	95.0	12641	18.3	178.9	4.3
6	1024	3754	72.6	9880	24.8	6517	13.6	5897	97.9	12714	15.9	181.1	4.1
7	1024	3755	72.6	10176	25.4	6556	15.1	5637	93.6	12726	20.0	181.0	4.6
8	1024	3729	72.2	10315	26.0	6614	12.7	5747	95.4	12738	16.7	180.2	4.4
9	1024	3769	72.8	9829	24.3	6501	15.7	5862	97.3	12715	16.9	181.5	4.5
10	1024	3824	73.9	9979	24.8	6379	12.5	5885	97.7	12717	16.0	179.8	4.0

A.2 Scatter plots of sample data



A.3 Statistical summary of sample data

filesystem access mode	ext2fs (kernel 2.3.0)		ext2fs (kernel 2.4.17)	
	block read	block write	block read	block write
repetitions	10	10	10	10
arithmetical mean	MB/s 12.8930	8.6499	13.1416	11.7567
geometrical mean	MB/s 12.8923	8.6499	13.1416	11.7556
harmonic mean	MB/s 12.8917	8.6498	13.1415	11.7545
median	MB/s 12.8462	8.6504	13.1455	11.7925
99% confidence interval for mean	MB/s 12.7833	8.6266	13.1140	11.6170
	MB/s 13.0026	8.6732	13.1692	11.8965
standard deviation	MB/s 0.1346	0.0286	0.0339	0.1716
coefficient of variation (C.O.V)	0.0104	0.0033	0.0026	0.0146
coefficient of skewness	0.5357	-0.0423	-0.7714	-1.2087
semi-interquantile range SIQR	MB/s 0.0975	0.0126	0.0212	0.0924
range	MB/s 12.7129	8.5996	13.0723	11.3877
	MB/s 13.1387	8.6982	13.1865	11.9238
max/min	1.0335	1.0115	1.0087	1.0471

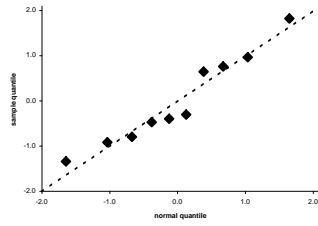
filesystem access mode	ext3fs (kernel 2.4.17)		ReiserFS (kernel 2.4.17)	
	block read	block write	block read	block write
repetitions	10	10	10	10
arithmetical mean	MB/s 12.4252	11.5818	12.4135	9.8006
geometrical mean	MB/s 12.4252	11.5805	12.4135	9.7985
harmonic mean	MB/s 12.4251	11.5791	12.4134	9.7963
median	MB/s 12.4336	11.5664	12.4180	9.8027
99% confidence interval for mean	MB/s 12.4040	11.4302	12.3916	9.6251
	MB/s 12.4464	11.7334	12.4353	9.9761
standard deviation	MB/s 0.0260	0.1861	0.0268	0.2155
coefficient of variation (C.O.V)	0.0021	0.0161	0.0022	0.0220
coefficient of skewness	-1.2036	0.0695	-2.1110	0.0054
semi-interquantile range SIQR	MB/s 0.0146	0.1497	0.0076	0.1918
range	MB/s 12.3701	11.3145	12.3447	9.5244
	MB/s 12.4541	11.8730	12.4395	10.0732
max/min	1.0068	1.0494	1.0077	1.0576

A.4 Normal quantile-quantile plots for sample data

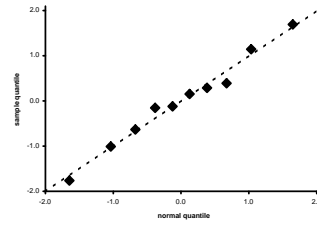
Note: The normal quantiles have been approximated using [Jai91, (12.1)]. The sample quantiles have been scaled to unit normal distribution ($y'_i = (y_i - \mu)/\sigma$).

ext2fs (kernel 2.3.0)

(a) block read

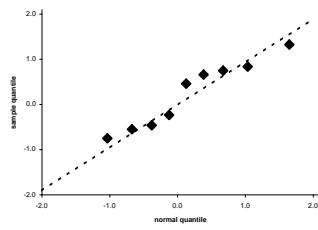


(b) block write

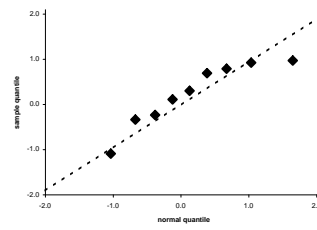


ext2fs (kernel 2.4.17)

(c) block read

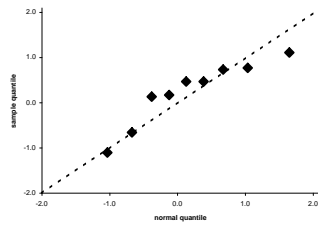


(d) block write

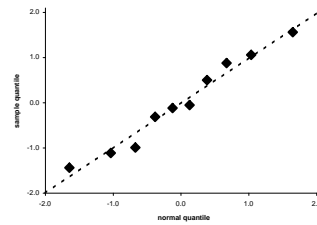


ext3fs (kernel 2.4.17)

(e) block read

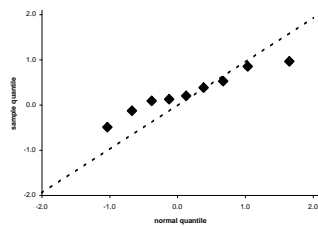


(f) block write

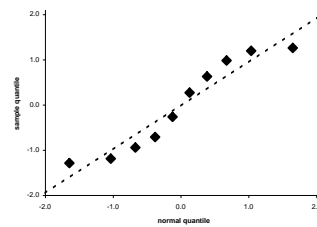


ReiserFS (kernel 2.4.17)

(g) block read



(h) block write



A.5 Two-factor full factorial design with repetitions

Note: The F-Table values in the ANOVA table ((c), rightmost column) have been approximated by $F_{[0.9,d,60]}$ because the value for $F_{[0.9,d,72]}$ was not available in tables. The unit of throughput values is MB/s.

(a) mean of observations

	block read	block write	row sum	row mean	row effect
<i>ext2fs (kernel 2.3.0)</i>	12.8930	8.6499	21.5429	10.7714	-0.8114
<i>ext2fs (kernel 2.4.17)</i>	13.1416	11.7567	24.8983	12.4492	0.8664
<i>ext3fs (kernel 2.4.17)</i>	12.4252	11.5818	24.0070	12.0035	0.4207
<i>ReiserFS (kernel 2.4.17)</i>	12.4135	9.8006	22.2141	11.1070	-0.4758
<i>column sum</i>	50.8732	41.7891	92.6623		
<i>column mean</i>	12.7183	10.4473		11.5828	
<i>column effect</i>	1.1355	-1.1355			

(b) interactions

	block read	block write
<i>ext2fs (kernel 2.3.0)</i>	0.9860	-0.9860
<i>ext2fs (kernel 2.4.17)</i>	-0.4431	0.4431
<i>ext3fs (kernel 2.4.17)</i>	-0.7138	0.7138
<i>ReiserFS (kernel 2.4.17)</i>	0.1709	-0.1709

(c) ANOVA Table

Component	Sum of Squares	Percentage of Variation	D.O.F.	Mean Square	F-Computed	F-Table (90%)
<i>y</i>	SSY 10907.61		80			
<i>y..</i>	SS0 10732.88					
<i>y - y..</i>	SST 174.73	100.00	79			
<i>access mode</i>	SSA 103.15	59.03	1	59.0347	232.677	2.79
<i>filesystem type</i>	SSB 36.25	20.74	3	6.9144	27.25	2.18
<i>interactions</i>	SSAB 3.41	1.95	3	0.6514	2.57	2.18
<i>errors</i>	SSE 31.92	18.27	72	0.2537		

(d) standard deviation of effects

<i>mean</i>	0.0563
<i>access mode</i>	0.5037
<i>filesystem type</i>	0.0563
<i>interaction</i>	0.0975
<i>error</i>	0.0975

(e) 90% confidence intervals for effects

Parameter	mean	
<i>mean</i>	11.4901	11.6754
<i>block write</i>	-1.2282	-1.0429
<i>block read</i>	1.0429	1.2282
<i>ext2fs (kernel 2.3.0)</i>	-0.9718	-0.6509
<i>ext2fs (kernel 2.4.17)</i>	0.7059	1.0268
<i>ext3fs (kernel 2.4.17)</i>	0.2603	0.5812
<i>ReiserFS (kernel 2.4.17)</i>	-0.6362	-0.3153

(f) 90% confidence intervals for interactions

	block read	block write
<i>ext2fs (kernel 2.3.0)</i>	0.8256 1.1465	-1.1465 -0.8256
<i>ext2fs (kernel 2.4.17)</i>	-0.6035 -0.2826	0.2826 0.6035
<i>ext3fs (kernel 2.4.17)</i>	-0.8743 -0.5534	0.5534 0.8743
<i>ReiserFS (kernel 2.4.17)</i>	0.0105 0.3314	-0.3314 -0.0105