Сравнение производительности raid1 – md, gflash, gmirror, zfs mirror

май 2015

В данном тестировании производится сравнение производительности алгоритмов балансировки в различных реализациях **raid1**.

Участники тестирования:

md raid1 - драйвер ядра **Linux** (md(4), mdadm(8) - Linux man page);
gflash - geom **FreeBSD** (github.com/geomflash/geomflash);
gmirror - geom **FreeBSD** (https://www.freebsd.org/doc/handbook/geom-mirror.html);
zfs mirror - реализация **raid1**в **zfs FreeBSD** (https://www.freebsd.org/doc/handbook/zfs.html).

Платформа для тестирования:

MB - Intel® Server Board S1200BTL;
CPU - Intel® Xeon® Processor E3-1230 (8M Cache, 3.20 GHz);
RAM - 16GB;
HDD - WD Black 2TB.
Диски для raid1 - 2 x WD Re 4TB (3,62TiB).

Операционная система:

Linux ubuntu 3.16.0-31-generic для **md raid1** (**I/O Scheduler** по умолчанию **deadline**);
FreeBSD 10.1-RELEASE-p5 для **gflash**, **gmirror**, **zfs mirror.**

Программа тестирования:

fio-2.1.11 (ioengine=libaio) для **md raid1**;
fio-2.1.9 (ioengine=sync) для **gflash**, **gmirror**, **zfs mirror** (выбор sync связан с реализацией **fio** в FreeBSD).

Тестирование выполняется с помощью **sh**-скрипта и включает в себя:
– идентификацию операционной системы и дисков для **raid1**;
– создание и отображение настроек **raid1**;
– последовательное выполнение заданий **fio** (всего 29 заданий);
– удаление и вставка диска входящего в **raid1** в процессе выполнения заданий **fio**;
– все данные тестирования записываются в файл для последующего анализа.

Для каждой реализации **raid1** создан свой **sh**-скрипт.

Все файлы тестирования доступны на github.com/gflash/test_raid1.

md raid1, **gflash**, **gmirror** являются реализациями **raid1** из блочных устройств. **zfs** является файловой системой со встроенным управлением блочными устройствами. Одной из возможных организаций пула **zfs** является зеркалирование (**mirror**) блочных устройств. Также **zfs** позволяет эмулировать поверх файловой системы блочное устройство (**zvol**). В связи с частым использованием **zfs** для предоставления блочного доступа интересно оценить производительность данного решения.

Использование **zvol** в качестве участника тестирования накладывает дополнительные условия и ограничения. При создании **zvol** для размещения метаданных необходимо дополнительное дисковое пространство. Также для нормального функционирования файловой системы, построенной по принципу **copy-on-write,** рекомендуется иметь не менее 10% свободного места на диске. Исходя из этого размер блочного устройства **zvol** выбран 3TiB (размер используемых дисков 3,62TiB). В **zvol** для того, чтобы чтение данных производилось с дисков необходимо сначала их записать. Для получения максимальной дисковой производительности операций чтения/записи в **zvol** отключены все дополнительные возможности **zfs**. Для исключения искажения результатов тестирования в оперативную память производится кэширование только метаданных. Настройки **zvol** отображены в регистрационном файле тестирования.

Далее описывается цель заданий **fio** и анализ полученных результатов.

Скорость последовательной записи большими блоками

Выполняется три задания последовательной записи 1TiB блоками по 128KiB:

```
2. --rw=write --bs=128k --iodepth=1 --offset=0t --size=1t
3. --rw=write --bs=128k --iodepth=1 --offset=1t --size=1t
4. --rw=write --bs=128k --iodepth=1 --offset=2t --size=1t
```

	md raid1			gflash			gmirror			zfs mirror		
	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%
2	166032	1297	0	166060	1297	0	166048	1297	0	126839	990	-24
3	152391	1190	0	152385	1190	0	152383	1190	0	112468	878	-26
4	130462	1019	0	130473	1019	0	130478	1019	0	96490	753	-26

Для **md raid1**, **gflash**, **gmirror** скорость записи прогнозируемо одинакова. Для **zfs mirror** скорость ниже приблизительно на 25%, что обусловлено дополнительными накладными расходами на обслуживание файловой системы.

Скорость случайной записи малыми блоками

Выполняется два одинаковых задания случайной записи в пределах 3TiB в течении 60 секунд блоками по 4KiB до и после заданий **№2**, **3**, **4**:

```
1. --rw=randwrite --bs=4k --iodepth=1 --filesize=3t --runtime=60
5. --rw=randwrite --bs=4k --iodepth=1 --filesize=3t --runtime=60
```

	md raid1			gflash			gmirror			zfs mirror		
	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%
1	720	180	0	716	179	-1	716	179	-1	38432	9608	5338
5	720	180	-1	708	177	-2	724	181	0	152	38	-79

Для **md raid1**, **gflash**, **gmirror** скорость записи в пределах погрешности. Очень показательны результаты для **zfs mirror**. Тест **№1** производился до записи 3TiB и запись случайных 4KiB-блоков благодаря **copy-on-write** преобразовалась в последовательную запись с огромной скоростью. Тест **№5** производился после записи 3TiB и использование **copy-on-write** не дало никаких преимуществ при записи случайных 4KiB-блоков, несмотря на достаточное количество свободного места в пуле **zfs**. Анализ статистики дисков входящих в **zfs mirror** показывает, что перед записью 4KiB блока производилось чтение 8KiB блока, модификация и затем его запись в свободную область. Также имелись накладные расходы на запись метаданных файловой системы. Скорость одинаковых тестов **№1** и **№5** отличается в 250 раз, а минимальная скорость (тест **№5**) почти на 80% ниже скорости остальных участников тестирования.

Скорость случайного чтения малыми блоками при различной глубине очереди

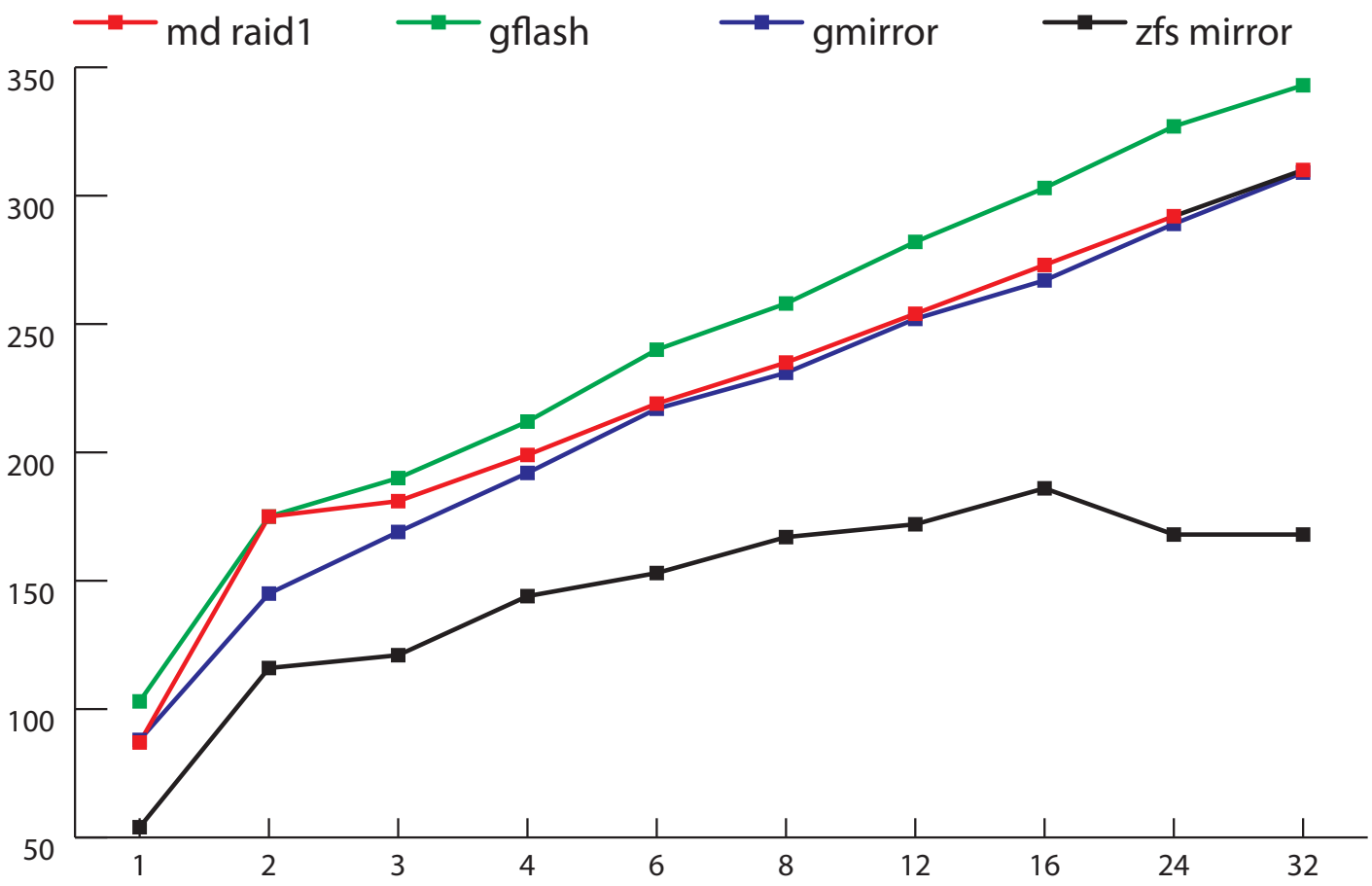
Выполняется десять заданий случайного чтения блоками по 4KiB в течении 60 секунд с различной глубиной очереди от 1 до 32:

```
6. --rw=randread --bs=4k --iodepth=1 --offset=10m --filesize=3t --runtime=60
7. --rw=randread --bs=4k --iodepth=2 --offset=11m --filesize=3t --runtime=60
```

```
8. --rw=randread --bs=4k --iodepth=3 --offset=12m --filesize=3t --runtime=60
9. --rw=randread --bs=4k --iodepth=4 --offset=13m --filesize=3t --runtime=60
10. --rw=randread --bs=4k --iodepth=6 --offset=14m --filesize=3t --runtime=60
11. --rw=randread --bs=4k --iodepth=8 --offset=15m --filesize=3t --runtime=60
12. --rw=randread --bs=4k --iodepth=12 --offset=16m --filesize=3t --runtime=60
13. --rw=randread --bs=4k --iodepth=16 --offset=17m --filesize=3t --runtime=60
14. --rw=randread --bs=4k --iodepth=24 --offset=18m --filesize=3t --runtime=60
15. --rw=randread --bs=4k --iodepth=32 --offset=19m --filesize=3t --runtime=60
```

	md raid1			gflash			gmirror			zfs mirror		
	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%
6	348	87	-16	412	103	0	352	88	-15	216	54	-48
7	700	175	0	700	175	0	580	145	-17	464	116	-34
8	726	181	-5	760	190	0	676	169	-11	484	121	-36
9	796	199	-6	848	212	0	768	192	-9	576	144	-32
10	876	219	-9	960	240	0	868	217	-10	612	153	-36
11	940	235	-9	1032	258	0	924	231	-10	668	167	-35
12	1019	254	-10	1131	282	0	1009	252	-11	688	172	-39
13	1094	273	-10	1215	303	0	1070	267	-12	744	186	-37
14	1171	292	-11	1308	327	0	1159	289	-12	672	168	-49
15	1242	310	-10	1373	343	0	1239	309	-10	672	168	-51

Для наглядности представим полученные данные в виде графиков:



gflash имеет самый производительный алгоритм балансировки, основанный на минимизации перемещений считывающих головок дисков, входящих в **raid1**. Только у **gflash** алгоритм балансировки даёт увеличение производительности при единичной очереди чтения относительно производительности одиночного диска.

md raid1 использует эмпирический алгоритм балансировки. Данный алгоритм имеет наилучшую эффективность при глубине очереди чтения равной двум. При других значениях глубины очереди чтения производительность в среднем на 8-10% ниже, чем у **gflash**.

gmirror имеет несколько алгоритмов балансировки, которые можно менять в процессе работы. Практический интерес с точки зрения производительности представляет только выбранный по умолчанию алгоритм **load**. Данный алгоритм также использует эмпирические правила. Его производительность немного уступает **md raid1** и в среднем на 12% ниже, чем у **gflash**.

zfs mirror имеет наихудшую производительность. Сложно судить об алгоритме балансировки, но характер кривой производительности до глубины очереди равной 16 очень напоминает **md raid1**. В среднем производительность чтения на 35-40% ниже, чем у **gflash**.

Производительность при нагрузке, характерной для баз данных

Выполняется случайное чтение (67%) и запись (33%) блоками по 8KiB в области размером 3TiB в течении 60 секунд при глубине очереди 32:

16.--rw=randrw --rwmixread=67 --bs=8k --iodepth=32 --offset=20m --filesize=3t --runtime=60

	md raid1			gflash			gmirror			zfs mirror		
	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%
16	1271	158	-13	1456	182	0	1406	175	-4	1153	144	-21
	616	77	-7	664	83	0	632	79	-5	496	62	-25

Лидирует **gflash**, незначительно отстает **gmirror**. Не совсем понятно отставание **md raid1** в скорости чтения. **zfs mirror** показывает худшие результаты, несмотря на равенство размера блока тестового задания и блока **zvol**.

Производительность последовательного чтения в несколько потоков

Выполняется последовательное чтение в один поток 8GiB блоками по 128KiB.

17.--rw=read --bs=128k --size=8g (offset=32g)

	md			gflash			gmirror			zfs		
	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%
17	173735	1357	0	170165	1329	-2	170635	1333	-2	129837	1014	-25

md raid1, **gflash**, **gmirror** - равная производительность в пределах погрешности измерения.

zfs mirror отстает на 25%, возможно это связано с фрагментацией **zvol** и дополнительными накладными расходами.

Выполняется последовательное чтение в два потока по 4GiB блоками по 128KiB. Чтение разнесено на начало и конец 3TiB тестируемой области.

18.--rw=read --bs=128k --size=4g (offset=16g)(offset=3040g)

	md			gflash			gmirror			zfs		
	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%
18	9411	73	-96	236020	1843	-1	238903	1866	0	128814	1006	-46

В **gflash** и **gmirror** алгоритм балансировки распределяет два потока чтения по двум дискам, входящим в **raid1**, что дает максимальную производительность.

zfs mirror отстает на 46%.

md raid1 проваливает тест. Отставание 96%. Алгоритм балансировки **md raid1** перемещает чтение потоков с одного диска на другой, что в итоге дает скорость случайного чтения.

Выполняется последовательное чтение в три потока по 2GiB блоками по 128KiB. Два потока читают в начале и один поток в конце 3TiB тестируемой области.

19.--rw=read --bs=128k --size=2g (offset=8g)(offset=12g)(offset=3030g)

	md			gflash			gmirror			zfs		
	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%
19	12710	99	-99	249909	1952	0	119274	931	-52	79685	622	-68

gflash имеет лучшую балансировку, что выливается в максимальную прозводительность.

В **gmirror** алгоритм балансировки не может распределить два близких потока чтения на один диск и дальний поток чтения на другой диск, в итоге отставание на 52%.

zfs mirror отстает на 68%.

md raid1 выдает производительность случайного чтения. Отставание 99%.

Выполняется последовательное чтение в четыре потока по 1GiB блоками по 128KiB. Два потока читают в начале и два потока в конце 3TiB тестируемой области.

20.--rw=read --bs=128k --size=1g (offset=2g)(offset=6g)(offset=3022g)(offset=3026g)

	md			gflash			gmirror			zfs		
	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%	KB/s	iops	%
20	86763	677	-60	219459	1714	0	128972	1007	-41	76235	595	-65

gflash уже традиционно, благодаря своему алгоритму балансировки, имеет максимальную прозводительность.

gmirror отстает на 41%.

md raid1 благодаря попарной близости потоков чтения, резко улучшает производительность случайного чтения. Отставание 60%.

zfs mirror отстает на 65%.

Производительность случайного чтения во время синхронизации

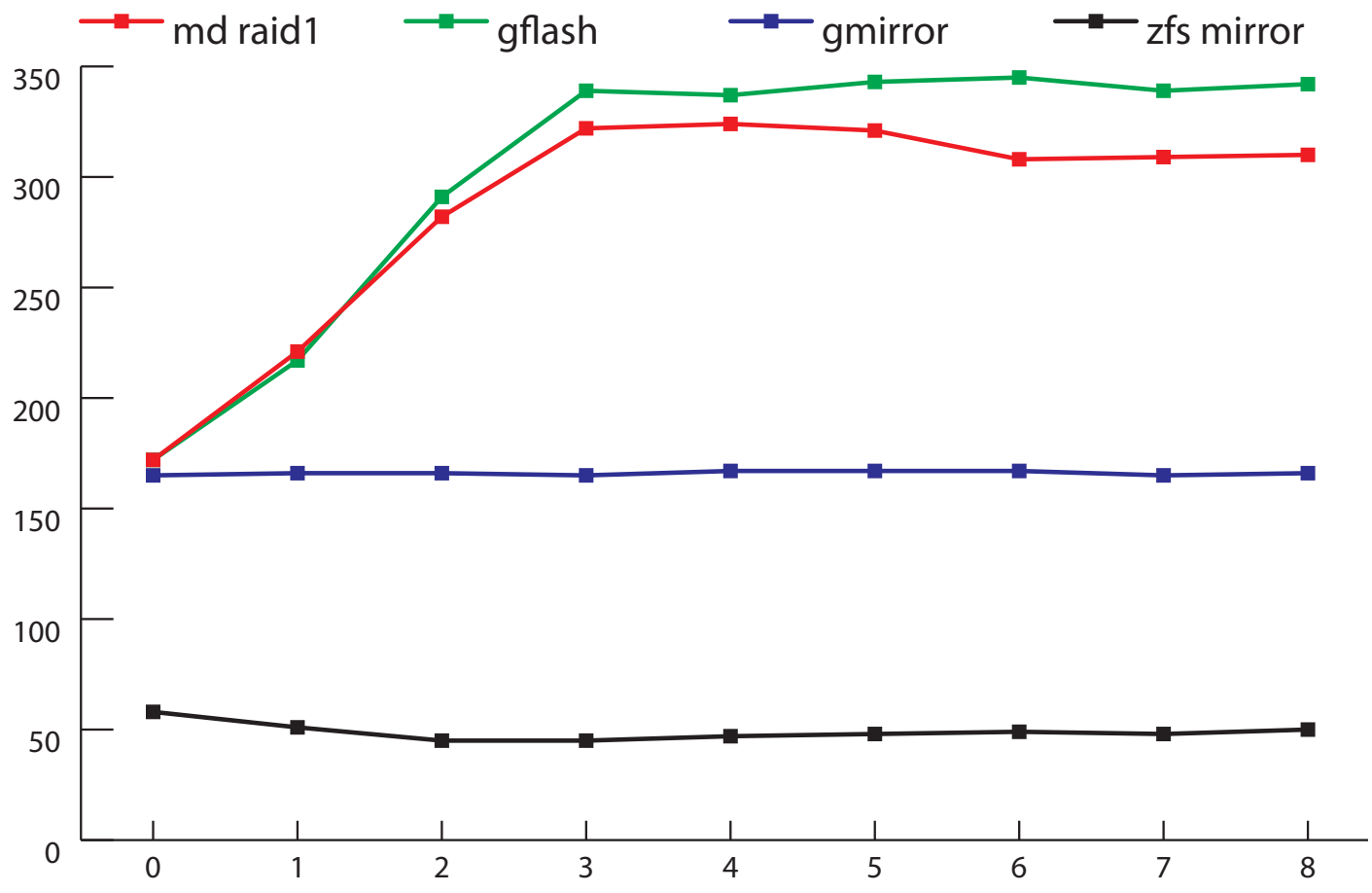
Во время процесса синхронизации девять раз с интервалом в 1 час выполняется случайное чтение блоками по 4KiB в области размером 3TiB в течении 60 секунд при глубине очереди 32:

```
21. --rw=randread --bs=4k --iodepth=32 --offset=1m --filesize=3t --runtime=60
22. --rw=randread --bs=4k --iodepth=32 --offset=2m --filesize=3t --runtime=60
23. --rw=randread --bs=4k --iodepth=32 --offset=3m --filesize=3t --runtime=60
```

```
24. --rw=randread --bs=4k --iodepth=32 --offset=4m --filesize=3t --runtime=60
25. --rw=randread --bs=4k --iodepth=32 --offset=5m --filesize=3t --runtime=60
26. --rw=randread --bs=4k --iodepth=32 --offset=6m --filesize=3t --runtime=60
27. --rw=randread --bs=4k --iodepth=32 --offset=7m --filesize=3t --runtime=60
28. --rw=randread --bs=4k --iodepth=32 --offset=8m --filesize=3t --runtime=60
29. --rw=randread --bs=4k --iodepth=32 --offset=9m --filesize=3t --runtime=60
```

	md			gflash			gmirror			zfs		
	sync, %	iops	%	sync, %	iops	%	sync, %	iops	%	sync, %	iops	%
21	0	172	0	0	172	0	0	165	-4	0	58	-66
22	16	221	0	15	217	-2	14	166	-25	8	51	-77
23	30	282	-3	30	291	0	27	166	-43	16	45	-86
24	45	322	-5	44	339	0	40	165	-51	24	45	-87
25	58	324	-4	58	337	0	53	167	-50	32	47	-86
26	71	321	-6	70	343	0	65	167	-51	40	48	-86
27	82	308	-11	82	345	0	76	167	-52	48	49	-86
28	92	309	-9	92	339	0	86	165	-51	55	48	-86
29	100	310	-9	100	342	0	95	166	-51	62	50	-85

Для наглядности представим полученные данные в виде графиков:



gflash и **md raid1** демонстрируют высокую скорость синхронизации и увеличение производительности чтения в процессе выполнения синхронизации. До выполнения 30% синхронизации производительность чтения примерно равна. Далее **gflash** увеличивает превосходство до 10%. В районе 40-70% выполнения процесса синхронизации у **md raid1** наблюдается пик производительности чте-

ния. Этот интересный эффект не связан с погрешностью измерений и является следствием чтения с половины одного диска (40-70% синхронизации) и чтения с полного объема (100%) у другого диска. Получается неожиданное подспорье для алгоритма балансировки **md raid1**.

gmirror имеет на 5% более низкую скорость синхронизации и в течении всего процесса синхронизации чтение выполняется только с одного диска. Также процесс синхронизации заметно снижает производительность чтения. В итоге имеем 50% отставание по производительности.

zfs mirror синхронизирует не диски, входящие в **raid1**, а данные файловой системы находящейся на дисках и называет этот процесс resilvering. По замыслу разработчиков **zfs** при малом проценте заполнения **zvol** это может сократить время синхронизации данных. В нашем случае выполнялась синхронизация 3TiB данных. Скорость синхронизации с учетом объема синхронизируемых данных в два раза ниже, что ставит под сомнение правильность выбранного решения. Производительность чтения в процессе синхронизации не возрастает, а даже несколько снижается. В итоге имеем 85% отставание по производительности чтения.

Выводы

Данное тестирование не может претендовать на абсолютную объективность, так как существует еще много нерассмотренных нагрузок и режимов работы, но позволяет оценить сильные и слабые стороны участников тестирования.

gflash

Обладает самым производительным алгоритмом балансировки. Предельно прост в конфигурировании и обслуживании.

md raid1

Эмпирические правила, заложенные в алгоритме балансировки, относительно хорошо справляются со случайным чтением и показывают неудовлетворительные результаты при последовательном чтении в более, чем один поток. К сильным сторонам можно отнести высокую производительность чтения во время синхронизации. Обладает самыми развитыми средствами конфигурирования и обслуживания.

gmirror

Имеет неплохую производительность. Алгоритм балансировки позволяет распределять последовательные потоки чтения по дискам один к одному. При увеличении количества последовательных потоков чтения эмпирические правила перестают справляться с оптимальным распределением нагрузки. Во время синхронизации чтение производится только с полностью синхронизированных дисков, что обуславливает низкую производительность чтения во время синхронизации. Имеет довольно много настроек.

zfs mirror

Относительно других участников тестирования имеет самую низкую производительность. Наверно это связано с накладными расходами на обслуживание файловой системы. Возможно использование встроенных возможностей, повышающих производительность (кэширование чтения в оперативную память и на **ssd**), несколько компенсируют потерю производительности, но использование **zvol** для предоставления блочного доступа с точки зрения производительности вызывает сомнения.