The Vinum Volume Manager

Greg “groggy” Lehey
LEMIS (SA) Pty Ltd
`grog@lemis.com`
`grog@FreeBSD.org`
`grog@NetBSD.org`

Adelaide, 16 January 2004

Vinum

The *Vinum Volume Manager* is an open source pseudo-device driver which:

- Implements virtual disk drives.
- Isolates disk hardware from the device interface.
- Maps data in ways which result in an increase in flexibility, performance and reliability.
- Implements the RAID-0, RAID-1, RAID-4 and RAID-5 models, both individually and in combination.
- Inspired by the VERITAS® volume manager, implements many of the concepts of VERITAS®.

Vinum availability

- Runs under FreeBSD.
- Imported to NetBSD -CURRENT.
- Ported to OpenBSD.
- Linux porting effort started.

Vinum RAID levels

Vinum implements many RAID levels:

- RAID-0 (striping).
- RAID-1 (mirroring).
- RAID-4 (fixed parity).
- RAID-5 (block-interleaved parity).
- RAID-10 (mirroring and striping), a combination of RAID-0 and RAID-1.

Vinum RAID levels (continued)

- Drive layouts can be combined to increase robustness, including striped mirrors (so-called “RAID-10”).
- In addition, other combinations are possible for which no formal RAID level definition exists.
- For example, combinations of RAID-5 and RAID-1, or dual RAID-5, are possible.
- Configuration can be changed on-line.

Foot protection

- Early volume managers emphasized reliability and performance, not ease of use.
- Many failures due to pilot error.
- GUIs tried to address this problem.
- Nothing beats understanding the concepts.
- Vinum supplies “easy to use” non-GUI interface.
- Hasn’t eliminated pilot errors.

Vinum objects

Vinum has a hierarchy of storage objects:

- The top level is the virtual disk or *volume*. Volumes effectively replace disk drives.
- Volumes are composed of one or more *plexes*.
- Each plex represents the total address space of a volume.
- Plexes provide redundancy.

Vinum objects (2)

Vinum has a hierarchy of storage objects:

- Plexes consist of one or more *subdisks*.
- A subdisk is a contiguous area of disk storage.
- Subdisks reside on Vinum *drives*.
- Drives are currently UNIX partitions.
- A drive can contain any number of subdisks.
- Each drive contains configuration information at the start.

Vinum features

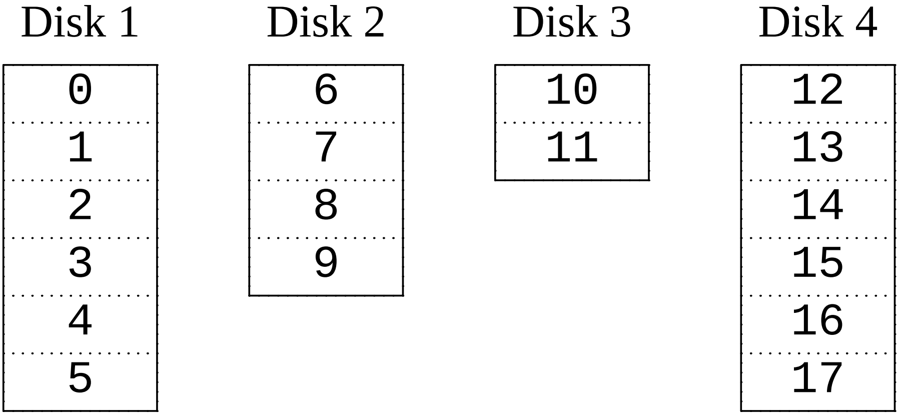
- Unlimited disk size: no intrinsic size limits.
- Faster access: concurrent access to multiple disks.
- Fault tolerance: disk failure does not mean volume failure.

Spreading the load

Vinum maps disk space to plexes in a number of ways:

- *Concatenated* organization accesses each subdisk in turn.
- *Striped* organization alternates between individual subdisks to spread the load.
- *RAID-4* and *RAID-5* mapping is a variant of striped organization.

Concatenated organization



Disk 1	Disk 2	Disk 3	Disk 4
0	6	10	12
1	7	11	13
2	8		14
3	9		15
4			16
5			17

Striped organization

Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23

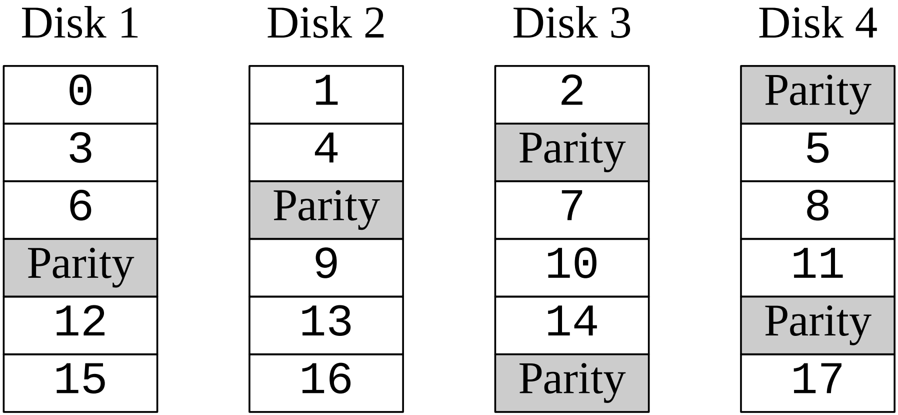
Redundant data organizations

- *Mirroring* (RAID-1) stores multiple copies of data.
- Vinum implements mirroring at the volume level.
- Each volume can contain up to 8 plexes.

Redundant data organizations (2)

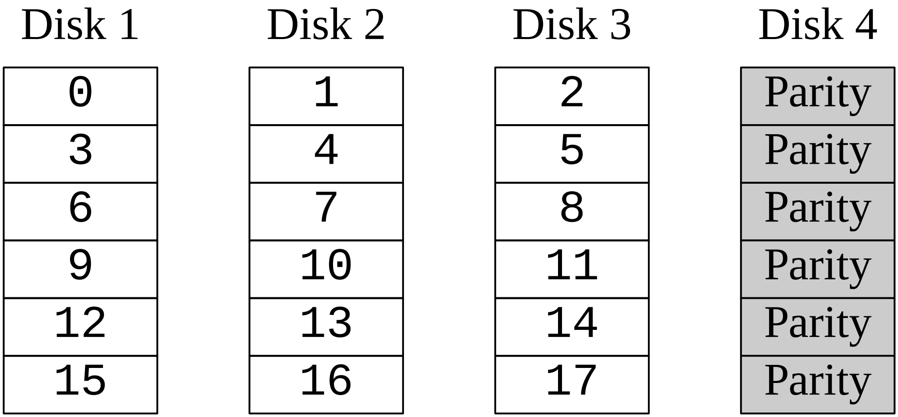
- RAID-4 and RAID-5 offer redundancy with lower storage overhead.
- Tradeoff: writes are much slower (about ¼ of read speed).
- Vinum implements RAID-4 and RAID-5 at the plex level.
- Use a variant of a striped organization with interspersed parity blocks.
- Storage overhead related to number of subdisks.

RAID-5



Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	Parity
3	4	Parity	5
6	Parity	7	8
Parity	9	10	11
12	13	14	Parity
15	16	Parity	17

RAID-4



Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	Parity
3	4	5	Parity
6	7	8	Parity
9	10	11	Parity
12	13	14	Parity
15	16	17	Parity

Comparing RAID-4 and RAID-5

- RAID-4 easier to implement than RAID-5.
- RAID-4 code a subset of RAID-5.
- Code almost identical:

```
/* subdisk containing the parity stripe */
if (plex->organization == plex_raid5)
  m.psdno = plex->subdisks - 1
     - (*diskaddr / (plex->stripesize * (plex->subdisks - 1)))
     % plex->subdisks;
else                                      /* RAID-4 */
                  m.psdno = plex->subdisks - 1;
```

- RAID-4 has performance issues when writing.
- RAID-4 has no corresponding advantages.
- Forget RAID-4.

Which plex organization?

Each plex organization has its own advantages:

- Concatenated plexes are the most flexible: can contain any number of subdisks, and the subdisks may be of different length.
- Striped (RAID-0) plexes reduce hot spots.
- RAID-5 plexes offer fault tolerance.
- Tradeoff for RAID-5: higher storage cost, slower write access.
- RAID-4 has no advantages over RAID-5.

Which plex organization? (2)

Plex type	Minimum subdisks	can add subdisks	must be equal size	application
concatenated	1	yes	no	Large data storage with maximum placement flexibility and moderate performance.
striped	2	no	yes	High performance in combination with highly concurrent access.
RAID-5	3	no	yes	Highly reliable storage, primarily read access.

Other Vinum features

- Online configuration via the *vinum* utility program.
- Automatic error detection and recovery where possible.
- State information for each object enables Vinum to function correctly even if some objects are not accessible.
- Persistent configuration enables automatic system start.
- Configuration includes state information.
  Degraded objects maintain state over a reboot,
  or even when moved to a new system.
- Support for Vinum root file systems.
- Online rebuild of objects.

Vinum configuration

- Vinum maintains a *configuration database* describing objects known to an individual system.
- Configuration created incrementally by *vinum(8)* from configuration files.
- The configuration file describes individual Vinum objects.

The configuration file

The definition of a simple volume might be:

```
drive a device /dev/da0s4h
volume myvol
  plex org concat
    sd length 1g drive a
```

This file describes a four Vinum objects:

- The `drive` line describes a disk partition (*drive*) and its location relative to the underlying hardware.
- Drive name is separate from device name, enabling drives to be relocated without confusion.
- The `volume` line describes a volume. The only required attribute is the name.

The configuration file (2)

- The `plex` line defines a plex for the preceding volume.
- Must specify organization (“`concat`”).
- Name defaults to *myvol.p0*.
- The `subdisk` line defines a subdisk for the preceding plex.
- Name defaults to *myvol.p0.s0*.

Creating the objects

After processing this file, *vinum(8)* produces the following output:

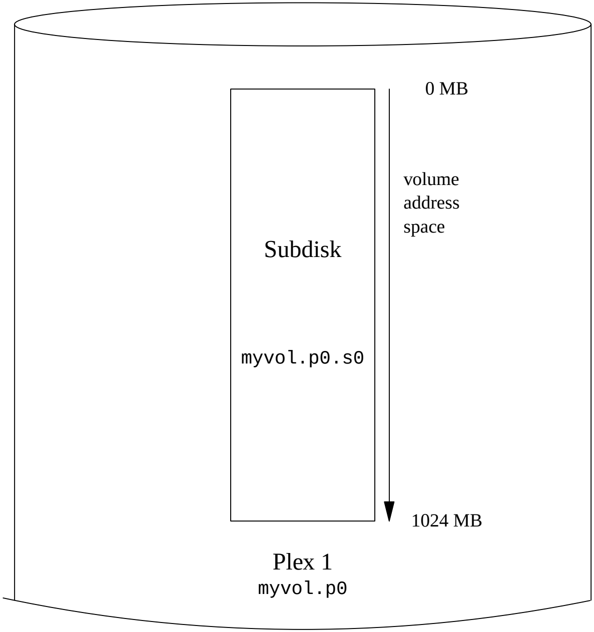
```
vinum -> create config1
1 drives:
D a                     State: up       /dev/da0s4h     A: 3070/4094 MB (74%)

1 volumes:
V myvol                 State: up       Plexes:       1 Size:       1024 MB

1 plexes:
P myvol.p0            C State: up       Subdisks:     1 Size:       1024 MB

1 subdisks:
S myvol.p0.s0           State: up       D: a            Size:       1024 MB
```

Graphical representation



Increased resilience: mirroring

To increase resilience, add plexes:

```
drive b device /dev/da1s4h
volume mirror
  plex org concat
    sd length 1g drive a
  plex org concat
    sd length 1g drive b
```

- Just define new objects.
- Drive *a* exists already, so it doesn't need redefinition.

mirroring (2)

After processing the definition, the configuration looks like:

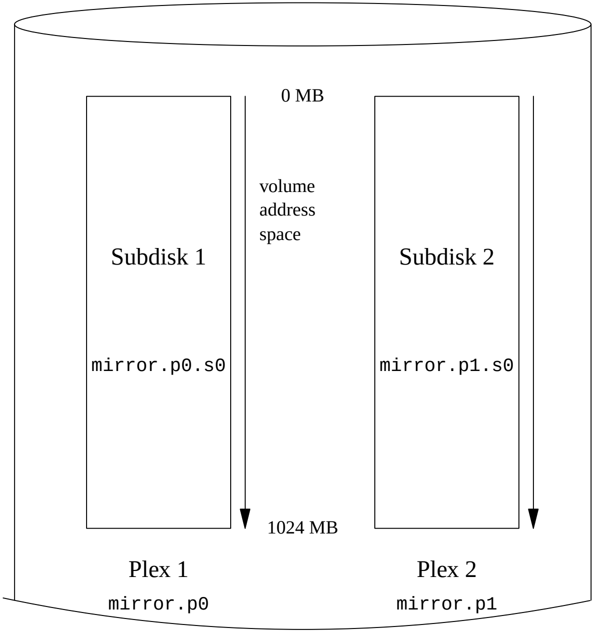
```
2 drives:
D a                     State: up       /dev/da0s4h     A: 2046/4094 MB (49%)
D b                     State: up       /dev/da1s4h     A: 3070/4094 MB (74%)

2 volumes:
V myvol                 State: up       Plexes:       1 Size:       1024 MB
V mirror                State: up       Plexes:       2 Size:       1024 MB

3 plexes:
P myvol.p0            C State: up       Subdisks:     1 Size:       1024 MB
P mirror.p0           C State: up       Subdisks:     1 Size:       1024 MB
P mirror.p1           C State: faulty   Subdisks:     1 Size:       1024 MB

3 subdisks:
S myvol.p0.s0           State: up       D: a            Size:       1024 MB
S mirror.p0.s0          State: up       D: a            Size:       1024 MB
S mirror.p1.s0          State: empty    D: b            Size:       1024 MB
```

Mirroring: graphical representation



What’s that fault?

- There’s only one problem with this volume: it’s faulty.
- The two plexes almost certainly contain different data, so Vinum can’t claim it’s the same.
- The correct method is to copy the data from one plex to another.
- This takes time.
- To cheat, specify the `setupstate` keyword when creating the volume.

Cheating

To have both plexes up on creation, specify:

```
volume mirror1 setupstate
  plex org concat
    sd length 1g drive a
  plex org concat
    sd length 1g drive b
```

- This tells Vinum to set the states of all plexes to up.
- It doesn’t work when adding plexes to a volume.

Optimizing performance

The mirrored volume in the previous example is more resistant to failure than an unmirrored volume, but its performance is less. Striping might improve performance:

```
drive c device /dev/da2s4h
drive d device /dev/da3s4h
volume stripe
  plex org striped 496k
    sd length 256m drive a
    sd length 256m drive b
    sd length 256m drive c
    sd length 256m drive d
```

Striping: Vinum output

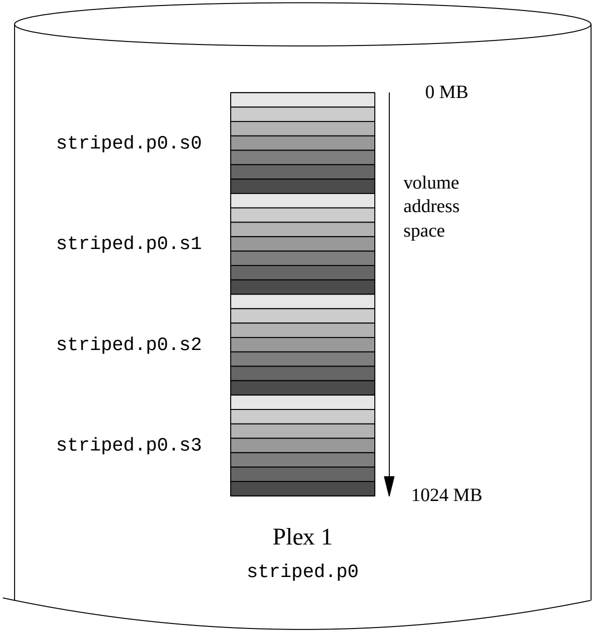
```
4 drives:
D a                     State: up       /dev/da0s4h     A: 1790/4094 MB (43%)
D b                     State: up       /dev/da1s4h     A: 2814/4094 MB (68%)
D c                     State: up       /dev/da2s4h     A: 3838/4094 MB (93%)
D d                     State: up       /dev/da3s4h     A: 3838/4094 MB (93%)

3 volumes:
V myvol                 State: up       Plexes:       1 Size:       1024 MB
V mirror                State: up       Plexes:       2 Size:       1024 MB
V stripe                State: up       Plexes:       1 Size:       1023 MB

4 plexes:
P myvol.p0            C State: up       Subdisks:     1 Size:       1024 MB
P mirror.p0           C State: up       Subdisks:     1 Size:       1024 MB
P mirror.p1           C State: faulty   Subdisks:     1 Size:       1024 MB
P stripe.p0           S State: up       Subdisks:     4 Size:       1023 MB

7 subdisks:
S myvol.p0.s0           State: up       D: a            Size:       1024 MB
S mirror.p0.s0          State: up       D: a            Size:       1024 MB
S mirror.p1.s0          State: empty    D: b            Size:       1024 MB
S stripe.p0.s0          State: up       D: a            Size:        255 MB
S stripe.p0.s1          State: up       D: b            Size:        255 MB
S stripe.p0.s2          State: up       D: c            Size:        255 MB
S stripe.p0.s3          State: up       D: d            Size:        255 MB
```

Striping: graphical view



Increased resilience: RAID-5

An alternative approach to resilience is RAID-5. A RAID-5 configuration might look like:

```
drive e device /dev/da4s4h
volume raid5
  plex org raid5 496k
    sd length 256m drive a
    sd length 256m drive b
    sd length 256m drive c
    sd length 256m drive d
    sd length 256m drive e
```

This plex has five subdisks, but its size is the same as the plexes in the other examples: the equivalent of one subdisk is used to store parity information.

RAID-5 (2)

After processing the configuration, the incremental configuration is:

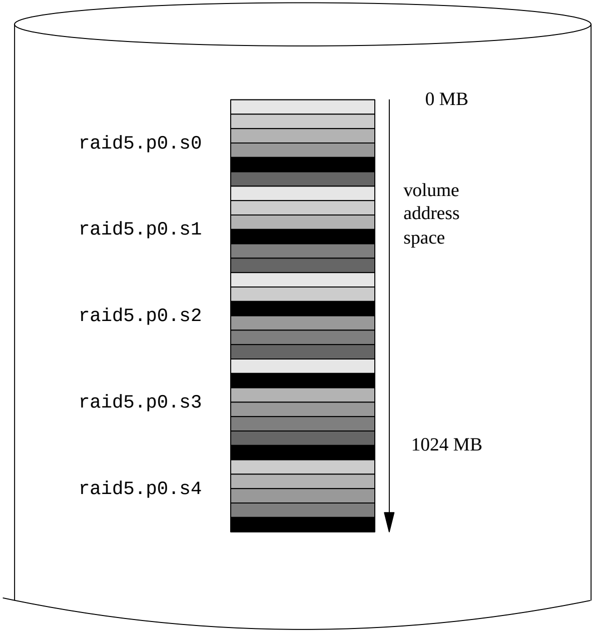
```
5 drives:
...
D e                     State: up       /dev/da4s4h     A: 3838/4094 MB (93%)

4 volumes:
...
V raid5                 State: down     Plexes:       1 Size:      1023 MB

5 plexes:
...
P raid5.p0           R5 State: init     Subdisks:     5 Size:      1023 MB

12 subdisks:
...
S raid5.p0.s0           State: empty    D: a            Size:       255 MB
S raid5.p0.s1           State: empty    D: b            Size:       255 MB
S raid5.p0.s2           State: empty    D: c            Size:       255 MB
S raid5.p0.s3           State: empty    D: d            Size:       255 MB
S raid5.p0.s4           State: empty    D: e            Size:       255 MB
```

RAID-5: graphical representation



RAID-5 (3)

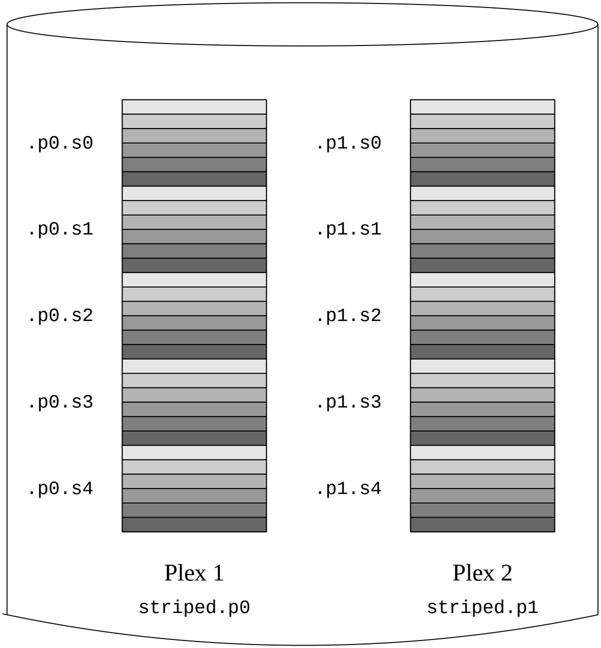
- On creation, RAID-5 plexes are in the *init* state.
- Must create parity before use.
- Initialization involves writing binary zeroes to all subdisks.

Resilience and performance

- With sufficient hardware, it is possible to build volumes which show both increased resilience and increased performance.
- Mirrored disks always give better performance than RAID-5.
- A typical configuration file might be:

```
volume raid10
  plex org striped 496k
    sd length 102480k drive a
    sd length 102480k drive b
    sd length 102480k drive c
    sd length 102480k drive d
    sd length 102480k drive e
  plex org striped 496k
    sd length 102480k drive c
    sd length 102480k drive d
    sd length 102480k drive e
    sd length 102480k drive a
    sd length 102480k drive b
```

Mirrored volume, graphically



Mirrored volume, config

After creating this volume, the incremental configuration is:

```
5 drives:
...

5 volumes:
...
V raid10                State: up       Plexes:       2 Size:        498 MB

7 plexes:
...
P raid10.p0           S State: up       Subdisks:     5 Size:        498 MB
P raid10.p1           S State: faulty   Subdisks:     5 Size:        498 MB

22 subdisks:
...
S raid10.p0.s0          State: up       D: a            Size:         99 MB
S raid10.p0.s1          State: up       D: b            Size:         99 MB
S raid10.p0.s2          State: up       D: c            Size:         99 MB
S raid10.p0.s3          State: up       D: d            Size:         99 MB
S raid10.p0.s4          State: up       D: e            Size:         99 MB
S raid10.p1.s0          State: empty    D: c            Size:         99 MB
S raid10.p1.s1          State: empty    D: d            Size:         99 MB
S raid10.p1.s2          State: empty    D: e            Size:         99 MB
S raid10.p1.s3          State: empty    D: a            Size:         99 MB
S raid10.p1.s4          State: empty    D: b            Size:         99 MB
```

Object naming

- Vinum assigns default names to plexes and subdisks.
- They may be overridden, but this is not recommended.
- Default plex names are derived by adding the characters `.p` and plex number to the volume name.
- Default subdisk names are derived by adding the characters `.s` and subdisk number to the plex name.

Devices

Vinum objects are assigned device nodes in the hierarchy */dev/vinum*. The configuration shown above would cause Vinum to create the following device nodes:

- The control devices */dev/vinum/control* and */dev/vinum/controld*, which are used by *vinum(8)* and the Vinum daemon respectively.
- Character device entries for each volume. These are the main devices used by Vinum. The configuration above would include the devices */dev/vinum/myvol*, */dev/vinum/mirror*, */dev/vinum/striped*, */dev/vinum/raid5* and */dev/vinum/raid10*.

Devices (2)

In addition, Vinum creates subdirectories for other objects.

- The directory */dev/vinum/plex* contains device nodes for each plex. The nodes have the name of the plex, for example */dev/vinum/plex/raid10.p1*.
- The directory */dev/vinum/sd* contains device nodes for each subdisk. The nodes have the name of the subdisk, for example */dev/vinum/plex/raid10.p0.s2*.

Creating file systems

- Vinum volumes are almost identical to disks.
- One exception: Vinum volumes do not contain a partition table.
- Confused some disk utilities, such as *newfs*.

On-disk configuration

- Configuration information on the drives has the same form as in the configuration files.
- Contains complete, explicit configuration information.
- Additionally, it contains information on object state.
- No entries for drives: they identify themselves.

Example configuration

```
volume myvol state up
volume bigraid state down
plex name myvol.p0 state up org concat vol myvol
plex name myvol.p1 state up org concat vol myvol
plex name myvol.p2 state init org striped 512b vol myvol
sd name myvol.p0.s0 drive a plex myvol.p0 state up len 1048576b driveoffset
 265b plexoffset 0b
sd name myvol.p0.s1 drive b plex myvol.p0 state up len 1048576b driveoffset
 265b plexoffset 1048576b
sd name myvol.p1.s0 drive c plex myvol.p1 state up len 1048576b driveoffset
 265b plexoffset 0b
sd name myvol.p1.s1 drive d plex myvol.p1 state up len 1048576b driveoffset
 265b plexoffset 1048576b
sd name myvol.p2.s0 drive a plex myvol.p2 state init len 524288b driveoffse
t 1048841b plexoffset 0b
sd name myvol.p2.s1 drive b plex myvol.p2 state init len 524288b driveoffse
t 1048841b plexoffset 524288b
sd name myvol.p2.s2 drive c plex myvol.p2 state init len 524288b driveoffse
t 1048841b plexoffset 1048576b
sd name myvol.p2.s3 drive d plex myvol.p2 state init len 524288b driveoffse
t 1048841b plexoffset 1572864b
```

Startup

- At startup, Vinum reads the configuration database from each drive.
- Sequence is reverse order of last modification time.
- Normally each drive contains an identical copy of the configuration database.
- After a crash, they may differ.
- After reading the most recent configuration, Vinum adds only objects which were not previously defined.

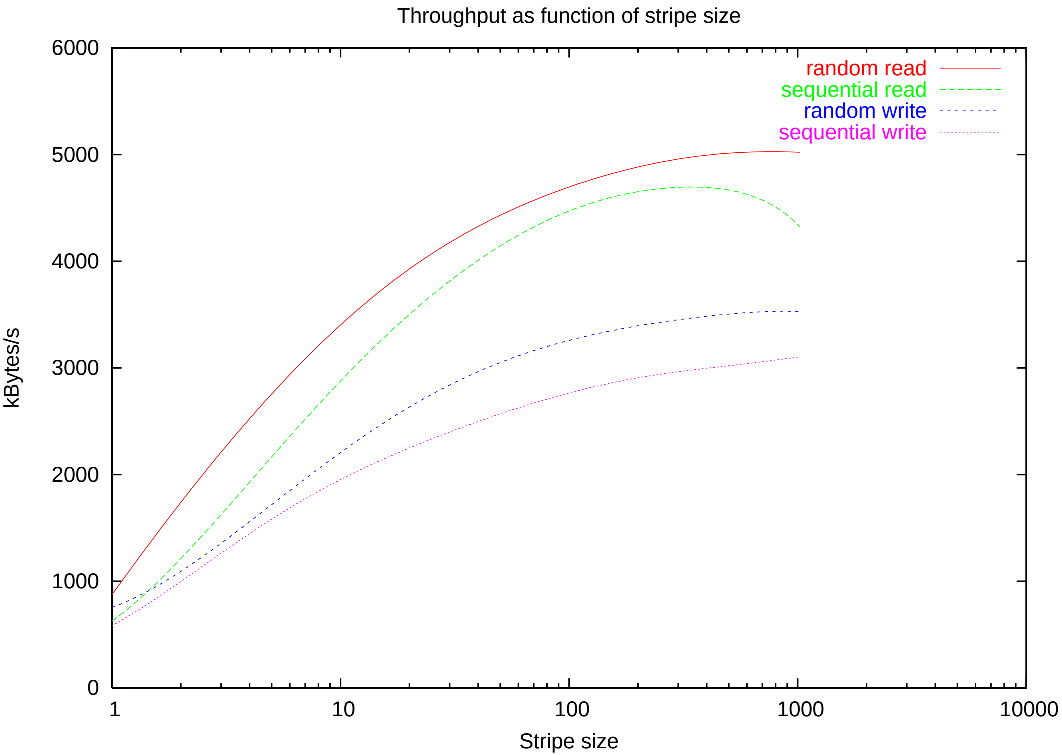
Performance

- Vinum overhead is negligible: performance is very close to corresponding hardware.
- The choice of requests has a strong impact on the overall subsystem performance.
- Some areas can be improved upon.
- Volume design can influence performance.

Stripe size

- In striped plexes, the stripe size has a significant influence on performance.
- Mapping requests can cause one high-level request to generate multiple I/O requests.
- This slows things down: increase in latency is the predominant factor.
- Spindle synchronization does not help: seek latency remains.
- Vinum does not transfer more than necessary.
- Therefore: choose a large stripe size (256 kB to 512 kB).

Performance vs. Stripe size



Performance vs. Stripe size

- Measurement used used eight concurrent processes to access volumes with striped plexes with different stripe sizes.
- Graph shows the disadvantage of small stripe sizes.
- File system structure influences stripe size: the relationship between cylinder group and stripe size can place the beginning of each cylinder group on the same subdisk.
- So: avoid stripe sizes which are powers of two.
- Disk cache size may also influence the transfer size.

RAID-1 mirroring

- Two different scenarios for these performance changes, depending on the layout of the subdisks comprising the volume:
- One disk per subdisk: optimum layout, both for reliability and for performance.
- Read access: by default, read accesses alternate across the two plexes. Performance improvement close to 100%.
- Write access: writes must be performed to both disks, doubling the bandwidth requirement. Available bandwidth is also double, so little difference in throughput.

Both plexes on same disks

- With less hardware, spread the subdisks of each plex over the same disks:

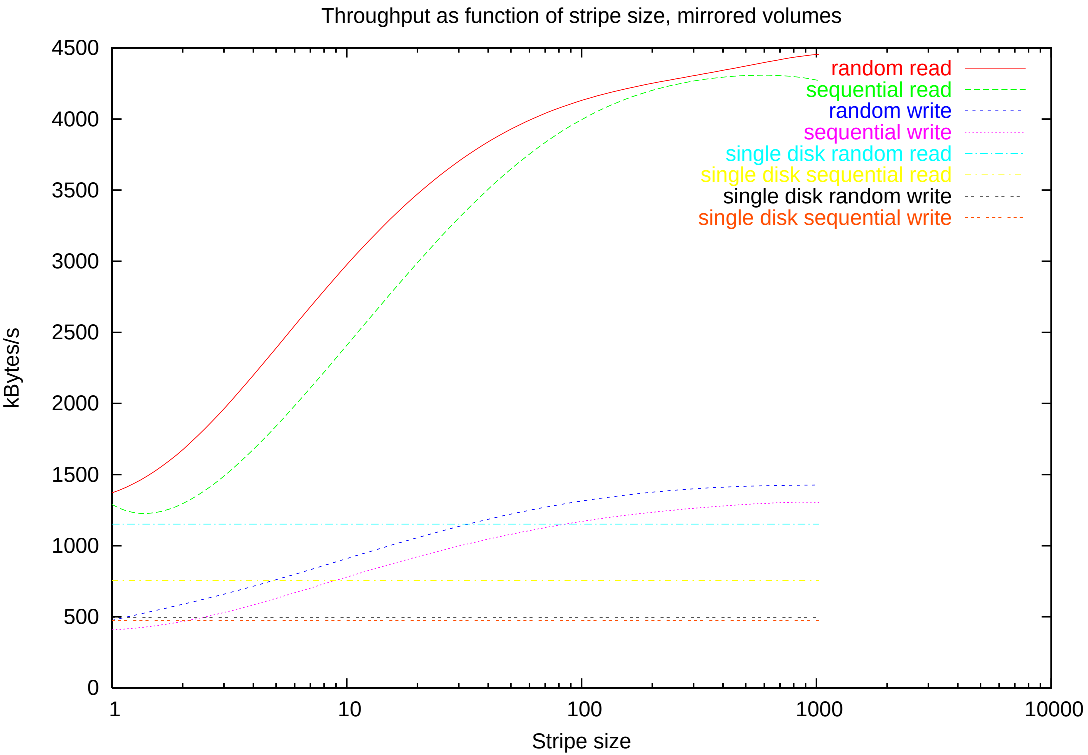
```
volume raid10
  plex org striped 512k
    sd length 102480k drive a
    sd length 102480k drive b
    sd length 102480k drive c
    sd length 102480k drive d
  plex org striped 512k
    sd length 102480k drive c
    sd length 102480k drive d
    sd length 102480k drive a
    sd length 102480k drive b
```

- Read access: by default, read accesses alternate across the two plexes. No increase in disk bandwidth, so little difference in performance through the second plex.

Both plexes on same disks

- Write access: writes must be performed to both disks, doubling the bandwidth requirement. Bandwidth has not increased, so write throughput decreases by approximately 50%.

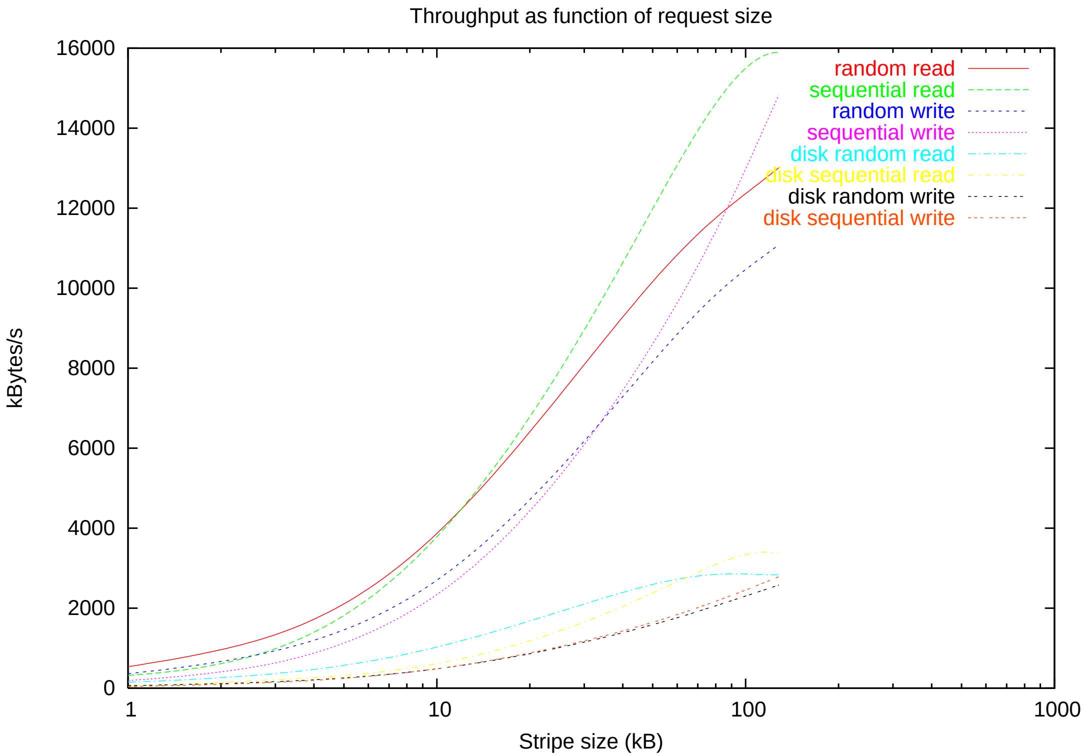
Both plexes on same disks (2)



Request size

- Throughput of a disk subsystem is the sum of the latency and transfer time.
- Latency is independent of transfer size.
- Overall throughput is strongly dependent on transfer size.
- Transfer size cannot be influenced, is in the range of 10 kB to 30 kB.

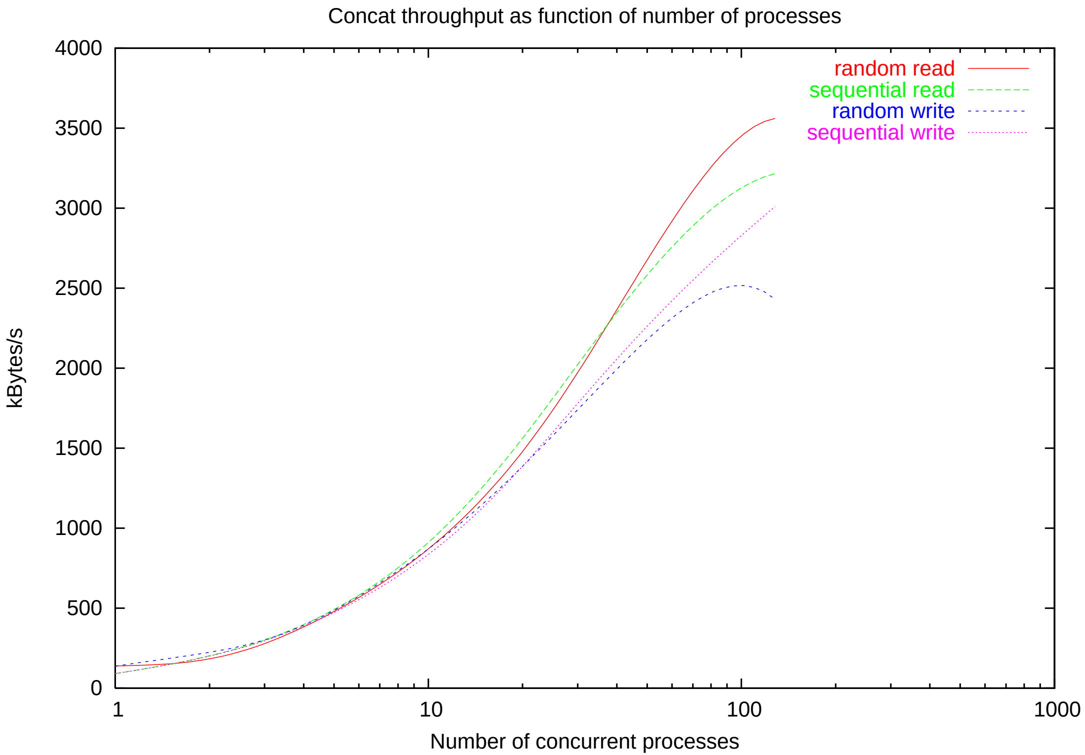
Request size



Concurrency

- Vinum aims to give best performance for a large number of concurrent processes performing random I/O.
- Means that single process transfers are not a good measure of throughput.
- Tests done with *rawio*, which creates multiple processes.
- Following graph ran with between one and 128 processes.

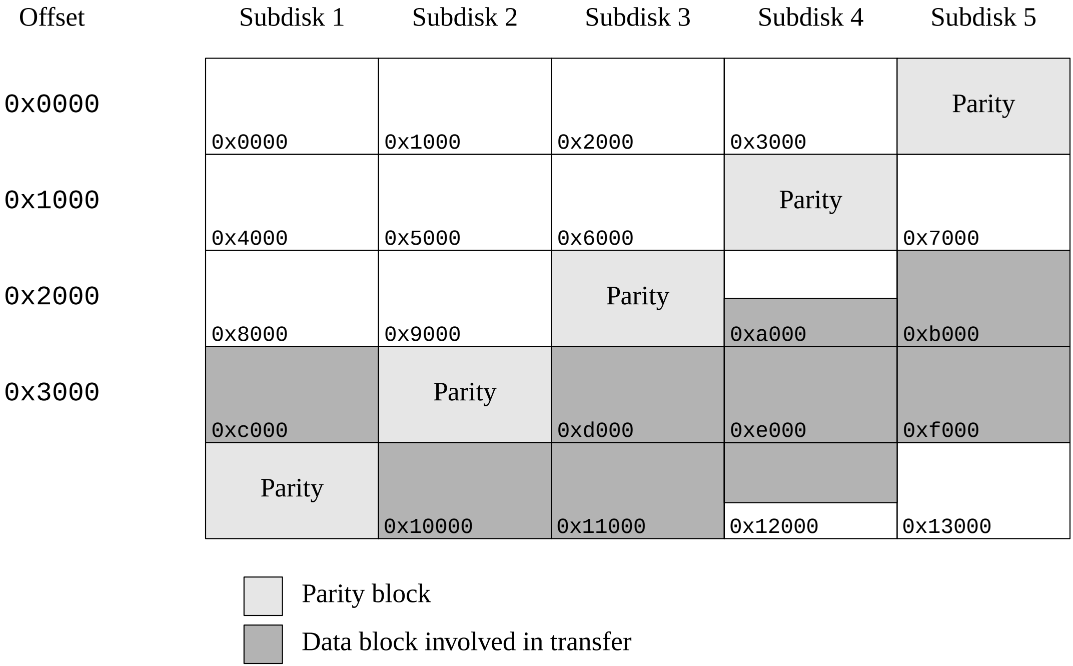
Concurrency (2)



Request structure

- For concatenated and striped plexes, Vinum creates request structures which map directly to the user-level request buffers.
- Only additional overhead is the allocation of the request structure.
- With RAID-5 plexes, the picture is very different.
- Algorithm works well when the total request size is less than the stripe width.

Request structure (2)



Offset	Subdisk 1	Subdisk 2	Subdisk 3	Subdisk 4	Subdisk 5
0x0000	0x0000	0x1000	0x2000	0x3000	□ Parity
0x1000	0x4000	0x5000	0x6000	□ Parity	0x7000
0x2000	0x8000	0x9000	□ Parity	▨ 0xa000	▨ 0xb000
0x3000	▨ 0xc000	□ Parity	▨ 0xd000	▨ 0xe000	▨ 0xf000
	□ Parity	▨ 0x10000	▨ 0x11000	▨ 0x12000	0x13000

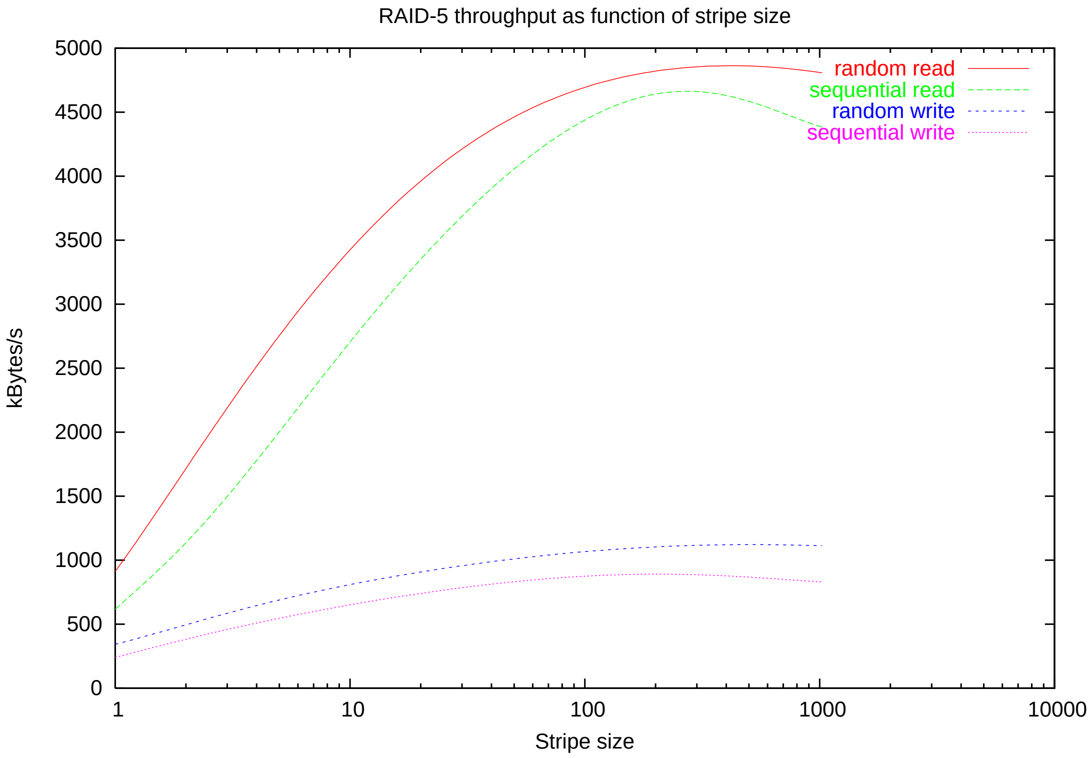
□ Parity block

▨ Data block involved in transfer

Request structure (3)

- Optimum approach performs a total of 5 I/O operations, one on each subdisk.
- Vinum treats this transfer as three separate transfers, one per stripe, and thus performs a total of 9 I/O transfers.
- This inefficiency should not cause any problems: the optimum stripe size is larger than the maximum transfer size.

RAID-5 throughput

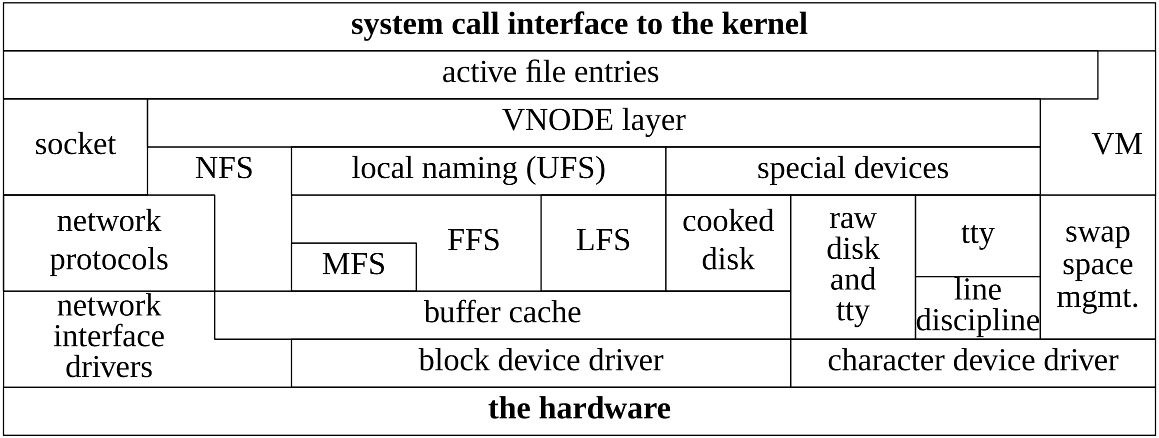


Implementation

- Vinum implementation required a number of tradeoffs.
- Vinum looks like a device, so it is accessed as a device.
- Instead of operating directly on the device, it creates new requests and passes them to the real device drivers.

Where Vinum fits

Without Vinum:



system call interface to the kernel

active file entries

socket

VNODE layer

VM

NFS

local naming (UFS)

special devices

network protocols

FFS

LFS

MFS

cooked disk

raw disk and tty

tty

line discipline

swap space mgmt.

network interface drivers

buffer cache

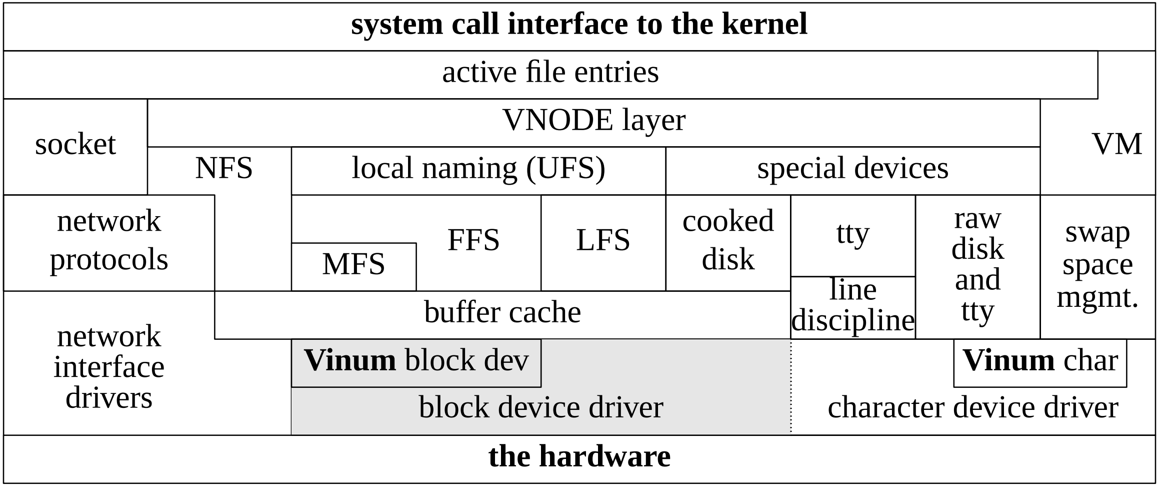
block device driver

character device driver

the hardware

Where Vinum fits (2)

With Vinum:



system call interface to the kernel

active file entries

socket

VNODE layer

VM

NFS

local naming (UFS)

special devices

network protocols

MFS

FFS

LFS

cooked disk

tty

line discipline

raw disk and tty

swap space mgmt.

buffer cache

network interface drivers

Vinum block dev

block device driver

Vinum char

character device driver

the hardware

Where Vinum fits (3)

- Diagram also shows gradual lack of distinction that has occurred between block and character devices.
- NetBSD implements disk block and character devices in the same driver.
- FreeBSD has completely dispensed with disk block devices, the shaded area in the figure.

Vinum under other OSs

- Vinum first appeared in FreeBSD in 1998.
- Ported to NetBSD in 2003 (two independent efforts).
- Now in NetBSD -CURRENT tree, still needs polishing.
- Ported to OpenBSD in 2003. Not in tree.
- Interest in Linux port revived in 2004 (last week).
- Join in!

More information

- More information about Vinum: `http://www.vinumvm.org/`.
- Sourceforge project: `http://sourceforge.net/projects/vinum/`.
- These slides: `http://www.vinumvm.org/papers/LCA2004/`.