GEOM
in infrastructure we trust

Paweł Jakub Dawidek
<pjd@FreeBSD.org>

AsiaBSDCon
Tokyo, 2008

History

• the GEOM framework first appeared in FreeBSD 5.0
• implemented by phk@
• sponsored by DARPA
• first commit in March 11, 2002
Nomenclature

- **class** – a kind of I/O transformation (e.g. mirror, stripe)
- **geom** – an instance of a class
- **provider** – provides storage (e.g. /dev/da0)
- **consumer** – connection between geom and provider
What is worth to remember

Access

- read
- write – can't be already open exclusively
- exclusive – can't be already open for write

```xml
<provider id="0x84af5d00">
  <geom ref="0x84af5d80"/>
  <mode>r3w3e7</mode>
  <name>ad0</name>
  <mediasize>160041885696</mediasize>
  <sectorsize>512</sectorsize>
  <config>
    <fwheads>16</fwheads>
    <fwsectors>63</fwsectors>
  </config>
</provider>
```
I/O requests

- **BIO_READ** – read data
- **BIO_WRITE** – write data
- **BIO_DELETE** – destroy/free data
- **BIO_FLUSH** – flush cache, put data onto stable storage
- **BIO_GETATTR** – ask about properties

GEOM on my laptop
**gconcat(8)**

- simple provider concatenation
- appears in /dev/concat/name
- usage:

  ```
  # gconcat label name da0 da1s1 da2s2d
  ```

```
<table>
<thead>
<tr>
<th></th>
<th>da0</th>
<th>da1s1</th>
<th>da2s2d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>
```

**gstripe(8)**

- RAID0
- appears in /dev/stripe/name
- usage:

  ```
  # gstripe label name da0 da1s1 da2s2d
  ```

```
<table>
<thead>
<tr>
<th></th>
<th>da0</th>
<th>da1s1</th>
<th>da2s2d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>
```
gmirror(8)

- RAID1
- appears in /dev/mirror/name
- autosynchronization
- usage:

```
# gmirror label name da0 da1s1 da2s2d
```

<table>
<thead>
<tr>
<th></th>
<th>da0</th>
<th>da1s1</th>
<th>da2s2d</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

graid3(8)

- RAID3
- appears in /dev/raid3/name
- bigger sector size
- \(2^a+1\) providers
- usage:

```
# graid3 label name da0 da1s1 da2s2d
```

<table>
<thead>
<tr>
<th></th>
<th>da0</th>
<th>da1s1</th>
<th>da2s2d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/0</td>
<td>2/0</td>
<td>1^2/0</td>
<td></td>
</tr>
<tr>
<td>1/1</td>
<td>2/1</td>
<td>1^2/1</td>
<td></td>
</tr>
<tr>
<td>1/2</td>
<td>2/2</td>
<td>1^2/2</td>
<td></td>
</tr>
<tr>
<td>1/3</td>
<td>2/3</td>
<td>1^2/3</td>
<td></td>
</tr>
<tr>
<td>1/4</td>
<td>2/4</td>
<td>1^2/4</td>
<td></td>
</tr>
</tbody>
</table>
**gjournal(8)**

- block-level journaling (not file system level journaling)
- file system independent
- can be used for file system journaling with minimal knowledge on FS side
- currently can be used for UFS journaling
- usage:

  ```
  # gjournal label da0
  # newfs -J /dev/da0.journal
  # mount -o async /dev/da0.journal /mnt
  ```

**ggatec(8), ggated(8)**

- exports storage over the network
- server usage:

  ```
  # echo "10.0.0.0/24 RO /dev/acd0" > /etc/gg.exports
  # echo "10.0.0.8/32 RW /dev/da1" >> /dev/gg.exports
  # ggated
  ```

- client usage:

  ```
  # ggatec create -o server /dev/acd0
ggate0
  # mount_cd9660 /dev/ggate0 /mnt/cdrom
  # ggatec create server /dev/da1
ggate1
  # newfs -J /dev/ggate1
  # mount /dev/ggate1 /mnt/data
  ```
**gshsec(8)**

- implements shared secret functionality
- all providers are needed to read the data
- appears in /dev/shsec/name
- usage:

  ```
  # gshsec label name da0 da1s1 da2s2d
  ```

<table>
<thead>
<tr>
<th></th>
<th>da0</th>
<th>da1s1</th>
<th>da2s2d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/0</td>
<td>1/0</td>
<td>2/0</td>
<td>3/0</td>
</tr>
<tr>
<td>1/1</td>
<td>1/1</td>
<td>2/1</td>
<td>3/1</td>
</tr>
<tr>
<td>1/2</td>
<td>1/2</td>
<td>2/2</td>
<td>3/2</td>
</tr>
<tr>
<td>1/3</td>
<td>1/3</td>
<td>2/3</td>
<td>3/3</td>
</tr>
<tr>
<td>1/4</td>
<td>1/4</td>
<td>2/4</td>
<td>3/4</td>
</tr>
</tbody>
</table>

**geli(8) 1/2**

- provides encryption and integrity verification
- utilizes crypto(9) framework – uses crypto hardware automatically
- supports various encryption algorithms (AES, camellia, blowfish, 3DES)
- supports various authentication algorithms (HMAC/md5, sha1, ripemd160, sha256, sha384, sha512)
- key can be split over serval components (passphrase, random bits from a file, etc.)
• allows to encrypt even root provider
• passphrase strengthened with PKCS#5v2
• two independent keys can be used
• starts as many worked threads as many CPU cores the system has

Comming soon (currently in perforce only):
• suspend/resume support

ZFS

• FreeBSD port implements two GEOM classes:
  • ZFS::VDEV – consumers-only class used to access GEOM providers
  • ZFS::ZVOL – providers-only class used for ZVOLs
geom(8) 1/2

- control utility for most GEOM classes
- few standard commands that work with all classes (list, status, load, unload)
- usage:

  # geom disk list
  # geom bsd status

geom(8) 2/2

- class-specific functions implemented via libraries (/lib/geom/)
- for the above, one can use g<class> command
- classes aware of geom(8):
  - cache
  - concat
  - eli
  - journal
  - label
  - mirror
  - multipath
  - nop
  - part
  - raid3
  - shsec
  - stripe
  - virstor
Questions?