The “Pairtree” Object Store

28 May 2008

John Kunze, California Digital Library

Pairtree

Thinnest possible smear on top of a file system to make an object system

cyocum

Platform-independent file hierarchy that factors the scarily hard repository into

- easy, powerful names and
- the stuff that's merely hard

Pairtree definition

A *pairtree* is a filesystem hierarchy mapping an identifier string into a unique object directory (or folder) location using pairs of characters

Assumptions:

- Any parts of the object not among its files (eg, metadata in a database) is secondary
- At that location are all the object's files and nothing but the object's files

Pairtree consequences

You can import a pairtree and, knowing *nothing* about object structure, can reliably

- Enumerate all objects and their identifiers
- Produce any object by requested id
- Maintain and back it up with ordinary OS tools
- Rebuild the collection in case of database corruption simply by walking the filesystem

Pairpaths

The *pairpath* algorithm maps successive bigrams of id into object's directory path

`abcdefg` ⇒ `ab/cd/ef/g/`

This is a *pairpath,* or ppath.

Every repository does something similar, so why not do it the same way?

Why pairs of characters?

Balances fanout (number of possible entries in any directory) and path depth

Compare

`ab2def3` ⇒ `ab/2d/ef/3/` `(36x36)`

to `ab2def3` ⇒ `a/b/2/d/e/f/3/` `(36)`

or `ab2def3` ⇒ `ab2def3/` `(36**7)`

Pairpath repositories permit limited sharing, but inside the object another story begins

Pairpath initialization

Pairtree version, root, and common prefix

```
./pairtree/
./pairtree/V1_0
./pairtree/root/
./pairtree/prefix
```

Example: prefix `http://n2t.info/` plus ppath

```
ur/n:/nb/n:/fi/-f/e2/00/71/01/5/
```

⇒ `http://n2t.info/urn:nbn:fi-fe20071015`

Pairpath termination

A *shorty* is a 1- or 2-char directory name

```
xy/          shorty, still in ppath
x/           shorty, still in ppath
xyz/         must be in object
foo.txt      file, must be in object
```

So, from the non-shorty directory "`cdef`" in

```
ab/cd/ef/cdef/
```

infer there's an object with identifier "`abcdef`"

Pairpath character conversion

But some chars can't appear in filenames

`/ \ : < > " | ? *`

So convert odd Windows & Unix chars to other, rarer filesystem-legal chars, or to hex-encoded values

`ark:/13030/xt12t34 ⇒`

`ar/k%/3a/=1/30/30/=x/t1/2t/34/`

Principle: no hidden riders

"Hidden riders" possibility when ppath doesn't terminate neatly in one directory; eg, when

```
ab/cd/foo/
ab/cd/bar/
ab/cd/readme.txt
```

Design choice: all must share identifier "abcd"

Declare improper "split" object, standard repair:

```
ab/cd/obj/{foo/,bar/,readme.txt}
```

What pairtree gives you

What's the thinnest smear we can add to our very well-understood filesystems and their universal tools (the universal "API") to create a very well-understood object storage substrate?

Hypothesis: pairtree is a simple generalized convention permitting a common approach

- that makes the rest smaller (eg, DSpace)
- and takes a big step towards object sharing

Where pairpath leaves off

Inside object is another story, such as, "*eflat*"

```
object/
  |  meta.txt
  |  files.txt
  |  data/
  |  v001/ . . . v004/
  |  meta/
  |  m001/ . . . . . . . m039/
  |  annotations/
  |  audits/
  |  config/
  |   . . .
  |  pairtree…
```