

SPE: Simple Policy Engine

Tom Haynes

thomas.haynes@sun.com

<http://blogs.sun.com/tdh>

Just what is spe?

- spe is a way to determine file layouts at creation
- Allows site admins to dictate where storage is created
- Allows clients to send hints to the server for layouts

Your pNFS is not my pNFS

- spe is not part of the spec
- spe does lay on top of the spec

Is there a need for spe?

- Were you paying attention during Tigran's presentation?
- Describe data
- Describe storage
- Describe cost
- What are they willing to pay (i.e., time) to create a file?

What do we know at creation?

- Basically, what is in the vnode...
- path
 - > /export/zfs/tdh/Documents/Pictures/family.jpg
- uid
- gid
- IP of client
- time

What don't we know?

- Expected file size
- Application
- Life duration
- Consumers

But can a data admin fudge those?

- /geo/WellBore/.../* .dat
 - > Wellbore data analysis
 - > path = /geo/WellBore && extension = dat
- /accounting/db/.../* .log
 - > Database log files
- /accounting/db/.../*
 - > Database files
- /usr/engr/scripts/*
 - > Small script files

Can be more complex

- subnet = floor && Time > 5PM
 - > build files
- subnet = floor && Time < 5PM
 - > User data
- id = 501 && extension = jpg
 - > Property shots for realtor
- id = 1918 && extension = jpg
 - > Porn

What is a policy?

- A tuple
 - > A list of npool(s)
 - > network pool - a collection of DS data sets
 - > # of data sets must be \geq stripe-count
 - > stripe-count
 - > unit-size
 - > rule
 - > An attribute expression which evaluates to TRUE or FALSE
- A data set can belong to only one npool
- A npool can belong to multiple policies

So what is a Policy Engine?

- Verification of policies before they are loaded
 - > Must have enough data sets
 - > Data set may be down
 - > Eventually it will come back up
- Systematic evaluation of policies to determine a match
- Selection of data sets to assign to storage
 - > 1st deliverable: Round Robin
 - > 2nd deliverable: Analysis of load

Some specifics

- spe is in user land
 - > Analysis engine will need floating point
 - > Want to be able to load new analysis engines
- policies are stored in SMF
 - > Persistent
 - > Redundant
 - > Clustered
 - > Hey, it is our paradigm!
- kernel uses doors to upcall
 - > usage data
 - > policy evaluation



The Policy Police - server side

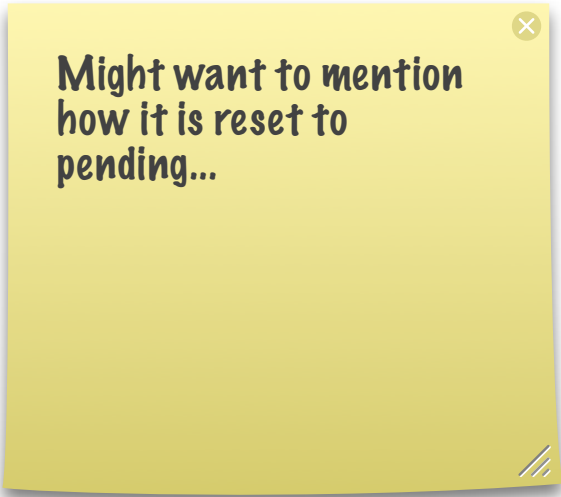
- A policy is only created if it is valid
- A policy is only modified if it stays valid
- When the daemon instantiates, if a persistent policy is invalid (or corrupt in SMF)
 - > We do not start up nfsd
 - > No data for you!

No data for you!

- Virgin system has the following:
 - > Default npool
 - > Initially empty
 - > Unassigned npool
 - > New DS data sets start here
 - > Default policy
 - > Initially empty
- The default policy must be valid for the mds to start up

Unassigned npool

- New data sets are grouped here
- State is pending
 - > “pending” authentication by admin
- State can be explicitly changed to accepted
- State is implicitly changed when data set is assigned to a npool
- State can be set to rejected



Might want to mention
how it is reset to
pending...

Default npool

- Always is assigned to default policy

The Policy Police - client side

- When the daemon instantiates and a policy is corrupt
 - > We log the problem
 - > We do not generate hints!

stripe-count min(4,ds)

unit-size will set after analysis

Setting up the default pool

```
# pnfsadm policy-get -s all
```

NAME	STRIPE_COUNT	UNIT_SIZE	NPOOLS	RULE
default	-	32k	default	-

```
# pnfsadm add -p default ds0:/pnfs/ds0,ds1:/pnfs/ds0
```

```
# pnfsadm status -npool default
```

NAME	MEMBER	MEMBER_STATE
default	ds0:/pnfs/ds0	ACCEPTED
default	ds1:/pnfs/ds0	ACCEPTED

```
# pnfsadm list -npool
```

NAME	MEMBER_COUNT	SIZE	USED	AVAIL	CAP
unassigned	10	5T	0	5T	0%
default	2	1T	0	1T	0%

```
# pnfsadm policy-get -s all
```

NAME	STRIPE_COUNT	UNIT_SIZE	NPOOLS	RULE
default	2	32k	default	-

Changing a property

```
# pnfsadm policy-set -s -o unit-size=8k default
# pnfsadm policy-get -s all
```

NAME	STRIPE_COUNT	UNIT_SIZE	NPOOLS	RULE
default	2	8k	default	-

Creating a new npool

```
# pnfsadm create films
# pnfsadm policy-create -s -o stripe-count=4,
unit-size=32k,npools=films,rule=path=/tank/films films
pnfsadm error: not enough associated datasets
# pnfsadm list -npool
```

NAME	MEMBER_COUNT	SIZE	USED	AVAIL	CAP
unassigned	10	5T	0	5T	0%
films	0	-	-	-	-
default	2	1T	0	1T	0%

```
# pnfsadm add -P films ds2:/pnfs/films,ds3:/pnfs/films,ds4:
/pnfs/films,ds5:/pnfs/films,ds6:/pnfs/films,ds7:/pnfs/films
# pnfsadm list -npool
```

NAME	MEMBER_COUNT	SIZE	USED	AVAIL	CAP
unassigned	4	2T	0	2T	0%
films	6	3T	0	3T	0%
default	2	1T	0	1T	0%

Creating a new policy

```
# pnfsadm policy-create -s -o stripe-count=4,
unit-size=32k,npools=films,rule=path=/tank/films films
# pnfsadm policy-get -s all
```

NAME	STRIPE_COUNT	UNIT_SIZE	NPOOLS	RULE
films	4	32k	films	path=/tank/films
default	2	8k	default	-

```
# pnfsadm create flix
# pnfsadm add -P flix ds8:/pnfs/films,ds9:/pnfs/films
# pnfsadm policy-set -s -o npools=films:flix films
# pnfsadm policy-get -s all
```

NAME	STRIPE_COUNT	UNIT_SIZE	NPOOLS	RULE
films	4	32k	films,flix	path=/tank/films
default	2	8k	default	-

Npools and Policies

```
# pnfsadm policy-create -s -o npools=flix:films,rule=uid=1066,
stripe-count=3,unit-size=16k -b dblogs tomper
# pnfsadm policy-get -s all
```

NAME	STRIPE_COUNT	UNIT_SIZE	NPOOLS	RULE
tomper	3	16k	flix,films	uid=1066
dblogs	2	64k	dblogs	ext=dbl
films	4	32k	films,flix	path=/tank/films
default	2	8k	default	-