afspy

Rechenzentrum Garching (RZG)
der Max-Planck-Gesellschaft

A quick update on the
highlevel python bindings

RZG site info

- RZG started as computing centre of IPP moving towards being the main computing centre of Max-Plank-Society
- provide HPC-machine MPG
- hosting and housing of linux cluster for different MPIs. (Also Tier2 for LCG)
- AFS losing importance

Present activities

- Retire TSM for HPSS as tape-backend
- Retire OSD onto disks or other services (iRODS, GPFS/GHI exported by HTTP and/or whatever)
- Mimic/sync AFS-group-management to UNIX-groups
- Others...

afspy - What's the goal ?

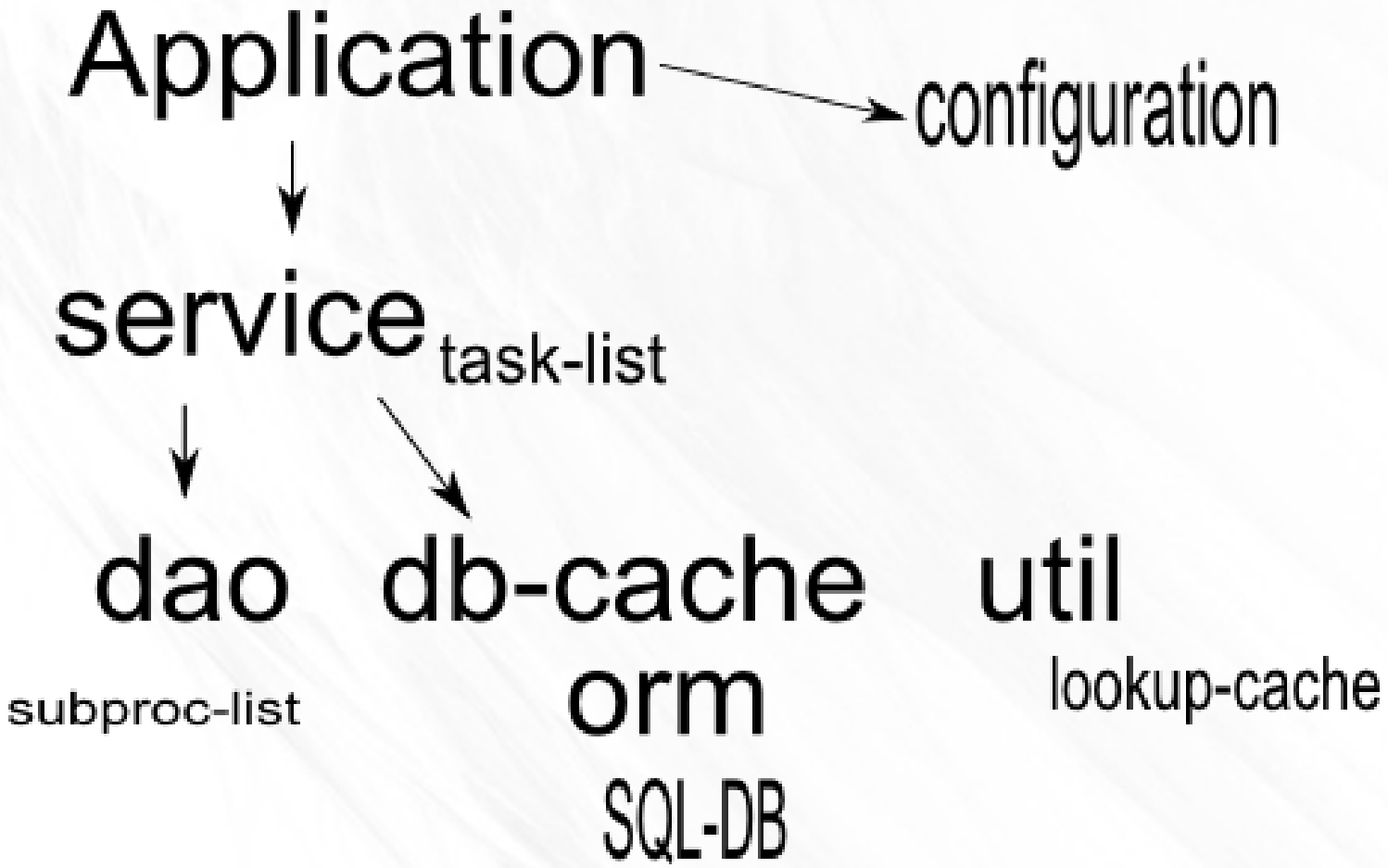
- previously we wanted to have
  both low- and high-level bindings

  → too complex.
- now : only high-level interface.
  - everyone can script „vos release“ in bash.

What's the goal ?

- pythonic: Everything is an object
- simple: Just call openafs-binaries and parse output
- quick: Transparent DB cache, keeps old data
- resilient: Asynchronuous execution of fully detached jobs in parallel

Structure


Application
configuration
service
task-list
dao
db-cache
util
subproc-list
orm
lookup-cache
SQL-DB

Structure

- services: interface to apps
- orm: db-caching and archiving of data
- dao: execution and parsing of openafs-commands
- utils: helper functions like DNS-lookup (internally UUID)

Services

- interface to apps.
- structured on logical units within openAFS.
  - Not related to single commands (like vos, bos etc....)
  - Usually a service uses more than one low-level command.
- a service returns an object or a list of objects

Logical Units

- Cell
- Bosserver (includes BNodes)
- Fileserver (includes partitions, volserver)
- DB-server
- VLDB / PTDB
- Volume

CellService

- returns Cell-Object
  - containing all relevant info about it:
  - e.g. hostnames of db-servers, fileserver

BosService

- read/modify stuff of a bosserver and it's BNodes :
  - Restart-times
  - Shutdown / startup
- get list of BNode-objects :
  - Create / delete
  - Start / stop
  - Status

FSService

- getting info about Fileserver and it's partitions
  - get list of partition-objects
  - get list of volume-objects of a partition

VolumeService

- get Volume object
- doing stuff with a volume:
  - create
  - remove
  - move
  - release
  - salvage

DBsService

- get DB-Server object :
  - Clone or not ?
  - IP-addrs
  - AFS-db types

PTDB / VLDB

- provides info about PTDB / VLDB :
  - All db-servers
  - Sync-site
  - DB-version
  - Sync-state

ProjectService

- provides a framework to organize volumes
- requires DB-Cache
  - A project has :
    - Name
    - Regex
    - Additional, ignored volumes
    - Specificity
    - Serverpartitions

Asynchronity

Each service is also a task-manager:

You can tell it to execute some method in a different thread.

Then, you need to poll it about the status.

Example:

> empty a fileserver by moving volumes in N threads in parallel.

DB-Cache

- transparent caching of objects implemented in services.
- keep old objects for statistics
- tested with mysql only, but should work with any DB supported by SQLAlchemy
- use-case : scout 2.0

Additional info stored in DB

- Projects
- Statistics
  e.g. Cell-Info:
  - Allocated space
  - stale allocated space

DAO

- called by a service.
  - execute standard afs-command
- either in place :
  - parse the output
  - return a partially filled object.

DAO - Resilience

- some AFS-commands have looong timeouts.
- execute them completely detached from the main process.

  Each DAO is also a sub-process manager.

  Service needs to poll it for results.

DAO

- called by a service.
  - execute standard afs-command
- fully detached:
  - return subprocess_identifier
  - stdout, stderr and rc stored in spool-files
  - service must poll DAO for result

Structure of an app

- create service-object
  - get data-objects
  - modify data-objects through service
- moving a volume :
  - get fileserver-object of dst from FSService
  - get volume-obj of src from VolumeService
  - use move method of VolumeService

afspy example

```
#!/usr/bin/env python


import afs

from afs.util.AFSConfig import parse_configs
from afs.service.FSService import FSService
from afs.service.VolumeService import VolumeService

import time


my_parser = argparse.ArgumentParser(parents = [afs.ARGPARSER], \
    add_help = False,epilog = afs.ARGPARSER.epilog)
my_parser.add_argument("--src_srv", required=True,\
    help="source-server")
my_parser.add_argument("--dst_srv", required=True,\
    help="destination-server")

#put all cmd-line options into afs.CONFIG
parse_configs(my_parser)


FSS = FSService()
VolS = VolumeService()

# get fileserver objects
src_fs = FSS.get_fileserver(afs.CONFIG.src_srv)
dst_fs = FSS.get_fileserver(afs.CONFIG.dst_srv)


# get list of all volumes on this fileserver
volume_list = FSS.get_volume_list(src_fs)

# move loop
move_count = 0
while move_count < len(volume_list) :
    if (VolS.active_tasks) >= 4 :
        time.sleep(10)

    VolS.move(volume_list[move_count], src_srv, dst_srv)

    # check finished moves
    for task in VolS.finished_tasks :
        # do something


# wait for last 4 moves to finish..
```

49,36 All

Outlook

- still have to implement a lot of this stuff,

  like the detached execution.

  And unit-tests.
- add circuit breaker into services to block on failing calls.
- write lots of example-apps

Question /Comments ?

Thank you!