FROM RESEARCH TO INDUSTRY

MILKCHECK
Distributed services manager

OLS | Aurélien Cedeyn <aurelien.cedeyn@cea.fr>

Outline

FROM RESEARCH TO INDUSTRY

NEEDS
PRESENTATION
ENGINE
USER INTERFACE
CONFIGURATION
USE CASES

NEEDS

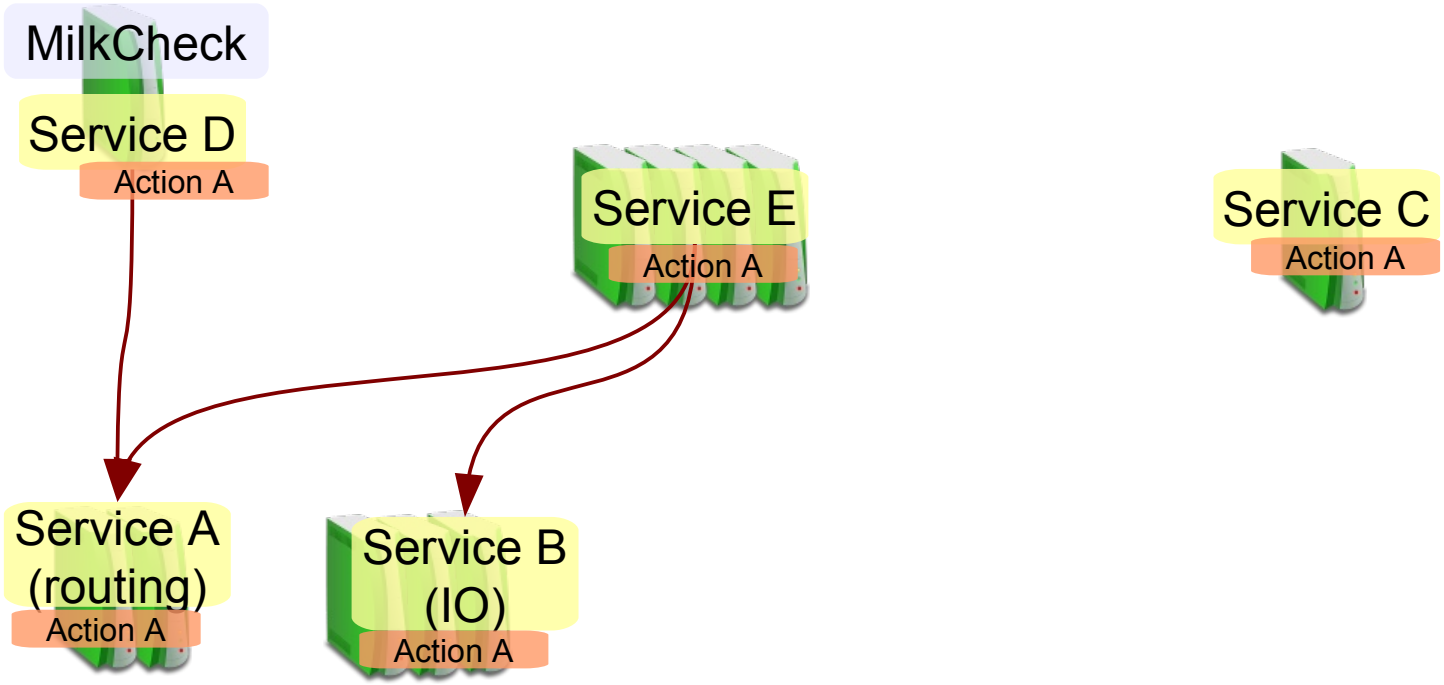
- For production management, manual actions often need to be done.
  - Start and stop local or distant services
  - Run on thousands of nodes
- Ease the support team job's
  - Have simple commands
  - Give a first diagnose of an issue
- This is often done with various home made scripts with classical drawbacks
  - Written by different admins
  - Do not have the same features
  - Often sequential
  - Hard to maintain
  - Difficult to scale up to thousands of nodes
- Write a tool to solve all of these issues: **Milkcheck**

PRESENTATION

- MilkCheck is a services manager, targeting:
  - Simple configuration
  - Compact output
  - Dependencies
  - Speed
- Python-based
  - Support any system with Python 2.4+
- Highly parallel
- Relies on ClusterShell
  - Python library
  - Introduced at OLS2012
  - Faster than pdsh

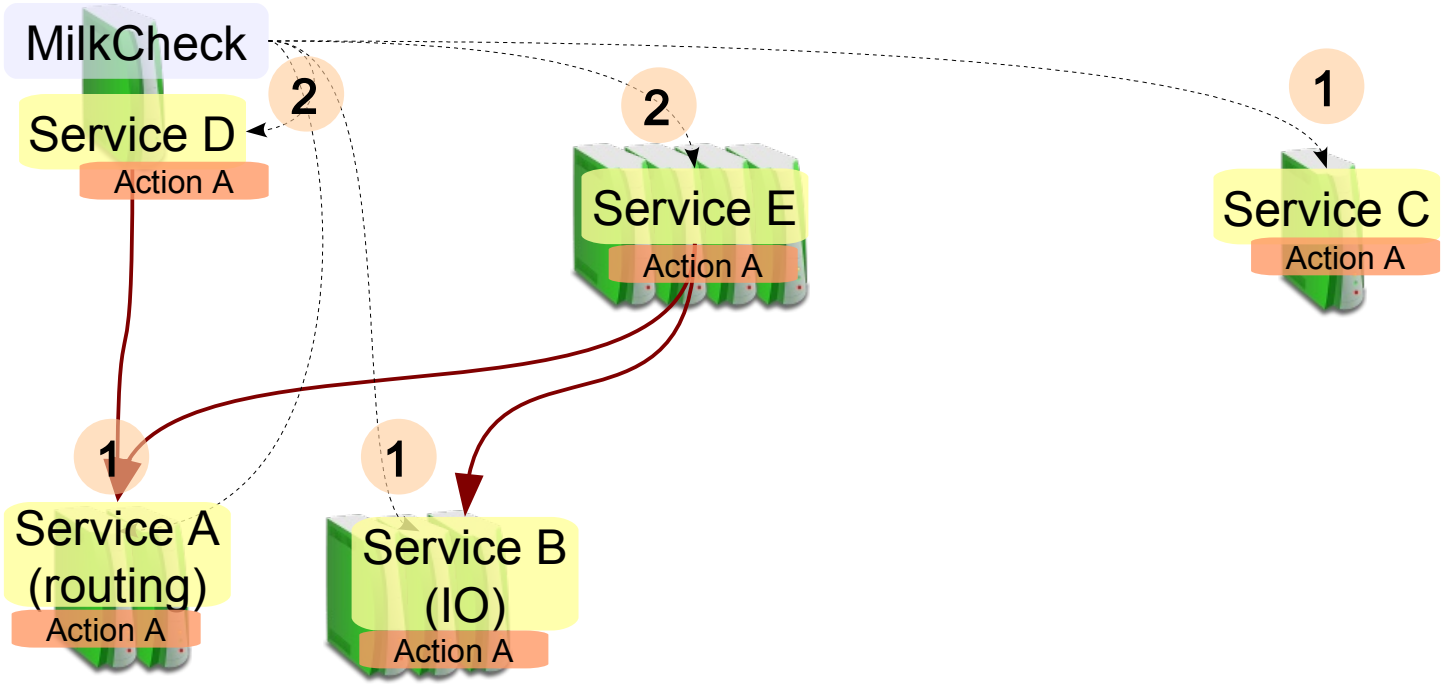
ENGINE (1/2)

- MilkCheck manages *services* with *actions*.
  - *Actions* run command locally or on **remote** nodes
  - *Services* can be linked each other with dependencies



ENGINE (2/2)

- MilkCheck runs actions as soon as possible, using maximum parallelism
  - Service A, B and C will be run immediately
  - Service D, depending on A will wait for its success before starting
  - Service E, depending on A and B, will wait for them too
  - Service C is independent



USER INTERFACE (1/2)

- Event-based
  - Tell exactly what is running

```
$ milkcheck status
ibms-master                                    [   OK   ]
[ccc_bridged,store,opensm-master]

Actions in progress
 > store.status on cloud[50-53,1500-1756,17... (2253)
```

Finished with success

Still running

Return is pressed
Display details on current running status

USER INTERFACE (2/2)

- Give more detail when error occurs
- Can display a summary, useful if running a lot of services

```
[root@cloud0 ~] # milkcheck status -s
ibms-master                                          [    OK    ]
ccc_bridged                                          [    OK    ]
opensm-master                                        [    OK    ]
lustre_servers.mgs                                   [    OK    ]
status lustre_servers ran in 0.27 s
 > cloud110: grep: /proc/fs/lustre/health_check: No such file or directory
 > cloud110 exited with 1
lustre_servers                                       [  ERROR   ]
scratch                                              [    OK    ]

 SUMMARY - 6 actions (1 failed)
 + lustre_servers.status
```

CONFIGURATION (1/3)

- Configuration is based on a collection of text files.
- Configuration files are YAML-based.

```yaml
services:
    local:
        actions:
            status:
                cmd: service crond status

    remote:
        target: node[1-1000]
        actions:
            status:
                cmd: service nfs status
```

- Services could be grouped

CONFIGURATION (1/3)

- Configuration is based on a collection of text files in a specified directory.
- Configuration files are YAML-based.

```yaml
services:
    local:
        actions:
            status:
                cmd: service crond status

    remote:
        target: node[1-1000]
        actions:
            status:
                cmd: service nfs status
```

milkcheck status

- Services could be grouped

CONFIGURATION (2/3)

- Services can have multiple actions, with any name
- Parallelism could be limited thanks to `fanout` option
- Each action could have its own `timeout`

```
services:
    switch:
        actions:
            status,check:
                timeout: 10
                cmd: ping -c1 switch-adm
    sshd:
        target: "@compute"
        fanout: 50
        require: switch
        actions:
            start,stop,status:
                cmd: service %SERVICE %ACTION
            check:
                cmd: pgrep sshd
```

CONFIGURATION (2/3)

- Services can have multiple actions, with any name
- Parallelism could be limited thanks to `fanout` option
- Each action could have its own `timeout`

```
services:
    switch:
        actions:
            status,check:
                timeout: 10
                cmd: ping -c1 switch-adm
    sshd:
        target: "@compute"
        fanout: 50
        require: switch
        actions:
            start,stop,status:
                cmd: service %SERVICE %ACTION
            check:
                cmd: pgrep sshd
```

milkcheck status
milkcheck check
milkcheck start
milkcheck stop

CONFIGURATION (3/3)

- Embedded scripts
  - Avoid deploying custom scripts on all remote nodes

```
services:
    no_user_process:
        actions:
            launch:
                target: foo[1-1000]
                cmd: |
                    ps h -e -o uid | sort -un | while read u; do
                        if [ $u -gt 1000 ]; then
                            killall -s SIGKILL -u $u
                        fi
                    done
```

```
$ milkcheck launch
no_user_process                                    [     OK    ]
```

USE CASES

- Use in production on our clusters
  - Managing up to 4000 nodes, 34 services
  - Start, stop and check all cluster services
  - Highly efficient thanks to parallelism

```
[root@cloud0 ~] # milkcheck status -s
ccc_bridged - CCC bridged service                                    [    OK    ]
ibms-master - IBMS service                                           [    OK    ]
ceanfs - CEANFS service                                              [    OK    ]
[filesystems]

Actions in progress
 > lustre_client.filesystems.status on cloud[50-53,70-71,121,136,1500-1756,17... (2253)
```

- Use in production on storage clusters
  - Check storage status
  - Monitor service status with a Nagios plugin
- Use in software testing
  - Ganesha

Thank you
Questions ?

https://github.com/cea-hpc/milkcheck

- ClusterShell at OLS2012
  https://github.com/downloads/cea-hpc/clustershell/ClusterShell_Paper_OLS2012.pdf
- ClusterShell vs Pdsh
  https://github.com/cea-hpc/clustershell/wiki/Pdsh

Commissariat à l'énergie atomique et aux énergies alternatives
CEA, DAM, DIF | 91297 Arpajon Cedex
T. +33 (0)1 69 26 40 00 | F. +33 (0)1 69 26 40 00

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019

CEA
DAM
DIF