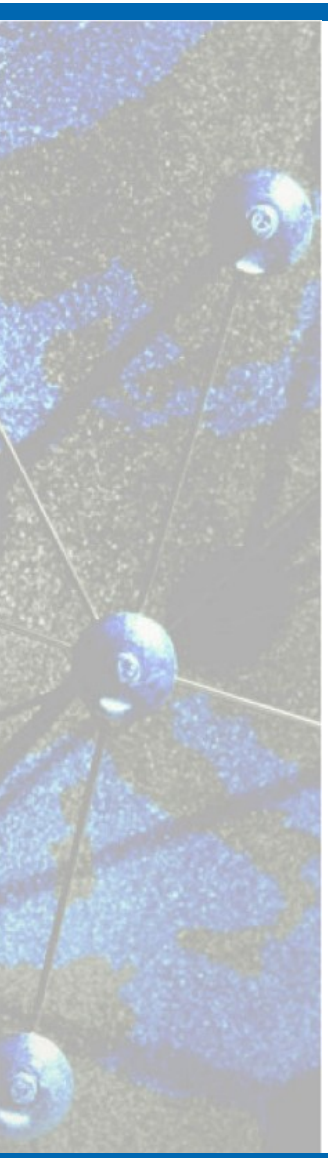

Some considerations about Application Accounting

Iván Díaz
Álvaro Simón

CESGA

Outline

Introduction

Codifying Application Data

Portal Interface problems

Why Application Accounting?

- Finer grained accounting and SLAs.
- Memory consumption and behavior (swapping and thrashing).
- Determination of failed executions.
- Provides feedback to developers about the use of their applications.

Detecting Application Use

- **By Executable:** Using the name of the executable or process.
- *Pros*
  - It does not need any user knowledge or participation.
  - Minimal client functionality also.
  - It can be done at kernel level with process accounting.
- *Cons*
  - Wrappers and name change in executables can make it invalid.
  - Needs to make a mapping beforehand (or impose a common format).
  - Version information normally not visible in the name.

Detecting Application Use (II)

- **By User Mapping:** There is a user or groups or users that map to a application. Done with some portals.
- *Pros*
  - It is generally transparent for the end user.
  - The application use is easy to completely determine.
  - Wrapping and portal interfaces can be used without problem.
- *Cons*
  - Determining the actual user of the resources can be problematic.
  - Versioning requires separate users, problems of granularity, aggregation.

Detecting Application Use (III)

- **By Role/group Mapping:** The application is codified in the role/group part of the DN. There is a user or groups or users that map to a application. Done with some portals.
- *Pros*
  - Preserves real user information AND application information.
  - Versioning easier to implement.
  - More intuitive.
- *Cons*
  - Users have to set role/group explicitly on sign-on.
  - It can conflict with other uses of groups and roles.

Versioning Problems

- Other problem can be different versions
- Separated statistics for versions that can be aggregated?.
- With what granurality it should be done?.

Interface Explosion

- There are 80+ views in the Accounting Portal.
- Application accounting is an orthogonal aspect applicable to all views.
- To reduce impact, it should be offered as a option, like grouping per region or per date.
- Possibly another tree branch of views could be required.

Are applications orthogonal?

- Applications are dependent in some part of VOs.
- So perhaps they could be subdivisions of VOs instead of a full fledged parameter.
- Versions could be a second subdivison.
- They could be implemented with the current VO view code or given an special interface.

Failed jobs

- Use of the exit status of processes to detect failures.
- Perhaps filtering trivial failures (file not found).
- Supplement computing data with ”failed” capability.
- Perhaps an additional checkbox?.


Thanks!