Use Cases

Jean Parpaillon
2011/05/31 10:32

Table of Contents

We collect here some example use cases.

Exercises for the Nice Meeting

Please, each partner, come to the Nice Meeting with a proposal of how you imagine we could write the system description, the user request, and the deployment plan for each of the uses cases given below.

Context

In Aeolus, we need to build three kind of 'languages':

- a **cloud description language**, that allows to express a **cloud description model** describing the components of the system (machines, packages and services) as well as the static structure of the relationships among them; this static system status should contain at least the following informations:
  - what service runs on what machine, depending on which other service, or software components;
  - what packages are on each machine, depending on which other packages;
  - which machines exist, located where, and connected to what other machine;
- a **cloud modification request language** allowing to express sophisticated modification to a cloud structure; this request language should be able to express modifications at all levels of the system (machines, packages, services), and to indicate non-functional requirements (like 'minimize downtime', 'minimize the cost of the additional rented machines', etc...)
- a **low level deployment language** able to describing modifications of the cloud by instructions like: stop service1 on machime m1; install p on machines m2 and m3; remove q from machine m4; start service1 on machine m5;

The solver should take a description of a cloud, a user request, and produce a plan, expressed in the low level language, that satisfies the user request.

Notice that a user request may imply a change in the set of available (virtual) machines, of their interconnection, the set of running services (with their dependencies), and the set of installed packages (with their dependencies).

Use cases

Here are a few use cases we can use as a first testbed in the Nice Meeting:

- Change the version of apache on a static web server farm (to fix a security issue). This use case needs service and updates to be properly synchronised.

Draft proposals for the **request language** in this use case:

forall m[let r =(apache,2.1) in Inst(r) and Service(r,m)=Running].
let r'=apache@m in Inst(r',2.2) and Service(r',m)=Running

Criticisms: there may be more then one version of a package installed, there may be more than one service provided by a single package.

Metadata needed: automaton describing the sequence of operations related to package installation and service status.

What about configuration files?

Draft proposal for the **cloud description information**:
(package dependencies, machine status(services and packages), automaton of service/package status transitions ... see example by Gigio on the blackboard)

Example of **computed deployment plan**:
any interleaving of (stop apache@m_i; remove apache@m_i; install apache=2.2@m_i; start apache@m_i) would satisfy the request (and thus is a possible output of the solver); but in this specific case, it is more efficient to pack all the operations related to the same machine (this is a nice heuristics in general, unless some external dependency forces to break this packing, for example if apache needs to wait for a new version of the MySQL server).

Extending the request with minimal SLA:

with #{m|Running(apache,m)} >= 5

- A CGI based web application (say phpbb3), which works well with either Apache or Cherokee, is running on an Apache server farm, which also have Cherokee installed ; a security critical bug has been reported on apache, and no fix is yet available; as a temporary countermeasure, the sysadmin wants to switch to Cherokee all the instances. This use case does not require installing new packages, only changing services.

- Deploy a PHP server farm, using PHP5, across at least 60 of the 137 (virtual) machines we have already rented on Amazon EC2, making sure not to use the machines on which PHP4 is already installed; a maximum of 20 extra machines may be rented if necessary; if machine geolocation is available, pool the PHP5 farm in the South European area. This use case may be used to experiment on the request language features.
- Database upgrade on a heavily loaded CMS, which has a static cache available (for example, Movable Type). Make Apache use the static cache, then upgrade the database, and make Apache go back to using the CMS. This use case may be used to experiment with service (re)configuration.
- Pulse2 Use-Case

nonfunctional requirements

Here are a few examples of nonfunctional requirements we might want to express:

- during the upgrade, we want to ensure, at any time, a minimal quality of service; for example: we want to have at least 20 running apache instances, at any time, no matter what (some old instances may be replaced one by one...; or we need to allocate temporarily 20 temporary instances)
- we want a plan that has 'short' inconsistent sections, to cope with potential failures
- add a 'cost' dimension to the request (how much I am ready to pay for each machine, and how much is the cost for each resource)
- possible 'online' behaviour of the planning algorithm : we may add new requests during the execution of the plan, and we would like to have an updated plan quickly

Resources

A few links to relevant resources for the Cloud.

- finnish report on the Cloud, 2010
- Ensemble in Ubuntu (see the repository on Lanuchpad and some example recipes)
- VMware Cloud CLI (ex SpringSource and now Cloudfoundry), and in particular see the VMC command line guide