Universitatea
Ştefan cel Mare
Suceava

An Architecture for Virtual Systems Management with Emphasis on Usability

eng. Gherman Ovidiu

Faculty of Electrical Engineering and Computer Sciences

Introducere (I)

- soluțiile existente dezvoltate în domeniul virtualizării acoperă o gamă largă de necesități;

- in problema managementului resurselor eterogene, cu platforme diverse conectate prin mecanisme de uz comun există o reală problemă în ceea ce privește planificarea execuției sarcinilor în funcție de fluxul de utilizatori și de nivelul de resurse existent;

- sistemele open-source trebuie configurate individual pentru a suporta o multitudine de resurse eterogene, planificarea sarcinilor făcându-se de obicei prin algoritmi triviali (de exemplu round-robin), fără a se ține cont de natura resurselor sau performanța acestora (sistemele comerciale sunt în general mai avansate);

Introducere (II)

- se propune o arhitectură care să îmbunătățească accesul la resursele eterogene și să permită utilizarea algoritmilor de planificare care pot ține cont de caracteristicile hardware/software sau de performanta ale elementelor componente;

- arhitectură va permite și implementarea unor facilități speciale pentru mediul de lucru (salvarea stărilor de lucru intermediare, migrarea sarcinilor fără repornire, capabilități predictive pentru administrarea resurselor, etc.);

- arhitectura se bazează pe tehnologii preexistente pentru a implementa elementele de bază;

- conceptul de virtualizare este utilizat pentru izolarea aplicației client de mașina fizică;

Module software

- resursele hardware ce suportă această tehnologie sunt de preferat (pentru performanțele sporite):
  - procesoarele ce implementează Intel VT-x, VIA VT sau AMD-V;
  - sistemele moderne (în special cele dedicate serverelor, dar și multe din sistemele de uz comun) implementează aceste facilități;

- la nivelul containerelor virtuale (infrastructura de virtualizare), opțiunile cele mai eficiente se bazează pe:
  - hipervizoarele de tip Xen sau Qemu;
  - containerele virtuale care pot fi administrate de un sistem (centralizat sau descentralizat) de administrare a mașinilor virtuale;

- un asemenea sistem este VirtualBox, care oferă numeroase avantaje, inclusiv in sistem de comenzi care permite administrarea programatică a mașinilor virtuale (creare/lansare/modificare);

Arhitectura propusă (I)

- sistemul este structurat pe mai multe nivele, fiecare având un rol specific:
  - infrastructura de virtualizare;
  - infrastructura de comunicații;
  - infrastructura de comandă și control;

- fiecare nivel este implementat prin module software preexistente, configurate corespunzător și controlate prin intermediul unui *middleware* comun;

Arhitectura propusă (II)

Infrastructura de virtualizare:

- utilizarea unui limbaj de scripting pentru administrarea serverelor (bash sau preferabil Perl) poate asigura un control eficient al acestor module;

- o dată instalate și configurate, pachetele hipervizor pot primi și lansa containere virtuale în mod automat;

- module dedicate pentru operațiuni specifice (lansarea mașinilor virtuale, oprirea acestora, mentenanța, monitorizarea și checkpointing-ul acestora) se pot implementa cel mai simplu direct în Perl, folosindu-se biblioteci dedicate controlului sistemelor virtuale (VirtualBox::Manage, Sys::Virt (libvirt));

Arhitectura propusă (III)

Infrastructura de virtualizare:

- modulele sunt controlate/raportează unui planificator central care monitorizează acest proces si implementează planificările sarcinilor de lucru ținând cont de:
  - •resursele avute la dispoziție;
  - •sarcinile existente;
  - •cerințele specifice ale utilizatorilor (dacă există).

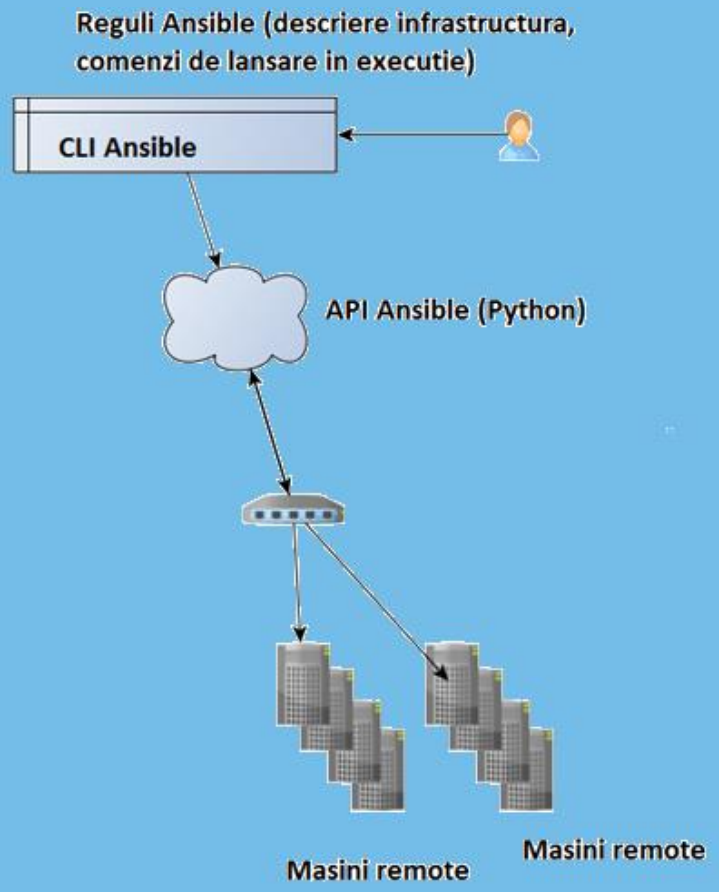
- modulele Perl pot de asemenea monitoriza resursele și pot extrage informații cu privire la:
  - •caracteristicile fizice ale acestora (HW/SW);
  - •cu privire la performanțele înregistrate și stărilor de lucru;

Arhitectura propusă (IV)

Infrastructura de virtualizare:

- toate informațiile sunt trimise periodic către planificator;
- acest lucru include monitorizarea conectivității și a stării de rulare a nodului prin intermediului unui *heartbeat* care să certifice că nodul este activ și operațional;
- se poate monitoriza starea curentă a sistemului;

Arhitectura propusă (V)



Infrastructura de comunicații (nivelul fizic):

- se bazează pe o biblioteca de comunicații existenta;
- modulele sunt ușor de integrate intr-o platforma software;

Arhitectura propusă (VI)

Infrastructura de comandă și control:

- planificatorul și nodurile-client (infrastructura de comandă și control) au rolul de a centraliza informațiile si de comanda lansarea containerelor virtuale;

- planificatorul primește informațiile de monitorizare a resurselor fizice precum și a utilizatorilor;

- planificatorul se ocupa si de autentificarea si managementul utilizatorilor;

- pe baza acestor informații se creează planificarea sarcinilor și fișierele de comandă care permit clienților (aplicații web) să lanseze sarcinile în execuție;

Arhitectura propusă (VII)

Infrastructura de comandă și control:

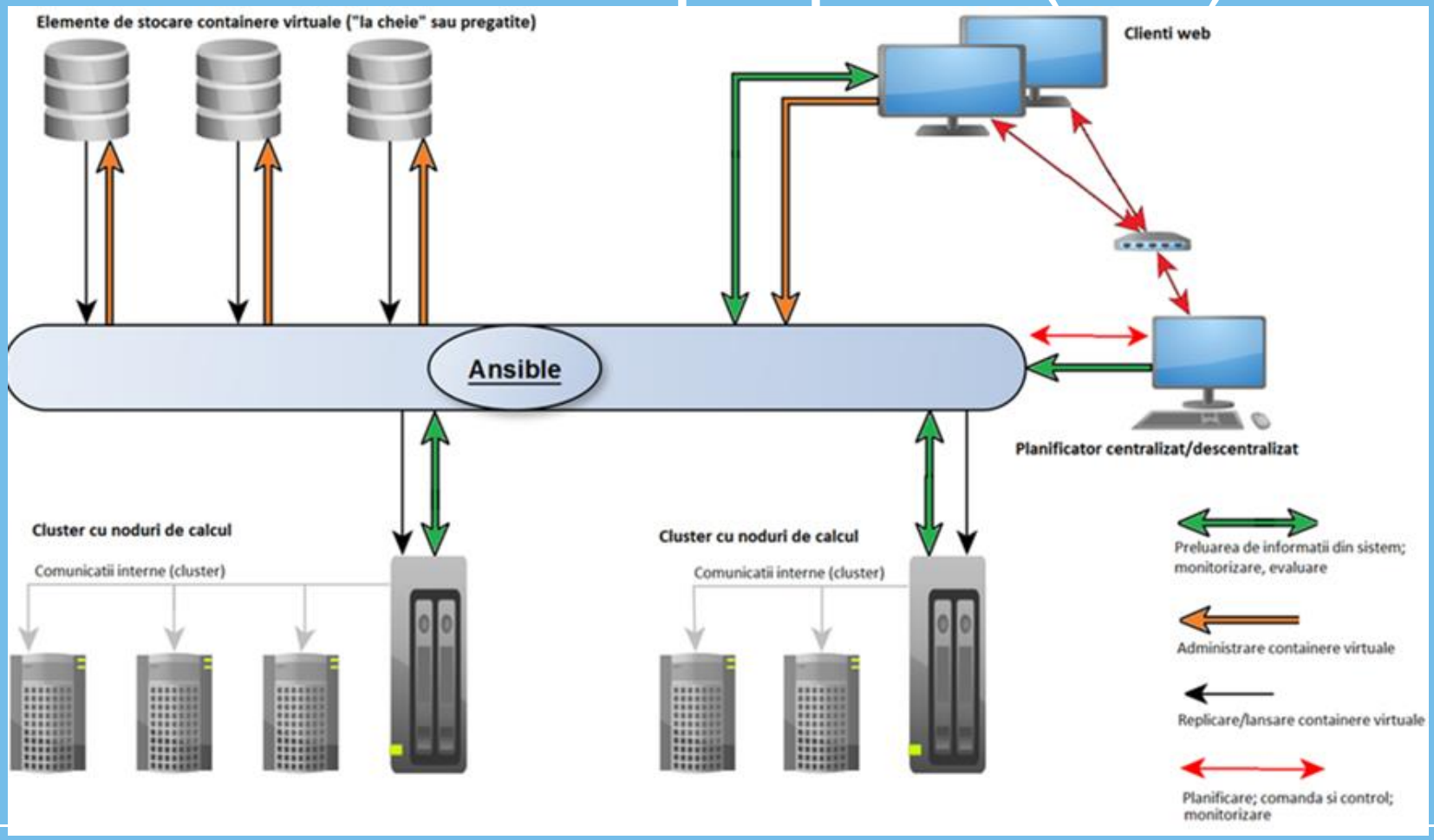
- clienții pot rula concurent, dar satisfacerea cerințelor acestora este strâns legată de disponibilul de resurse hardware și software;

- dacă cererea este mai mare decât oferta, clienții pot fi prioritizați în funcție de anumite criterii (inclusiv explicit, pe baza datelor de login, dacă se dorește) și puși în coada de așteptare;

- coada poate fi re-creata daca planificatorul consideră necesar pentru optimizarea planului de execuție;

- informațiile colectate precum si rezultatele analizei acestora vor putea fi reprezentate grafic si numeric, la cerere, de către clientul web sau planificator, prin intermediul unei biblioteci care va fi stabilită ulterior;

Arhitectura propusă (VIII)



Arhitectura propusă (IX)

- arhitectura nu își propune să ofere garanții de execuție, ci funcționează pe principiul „*as is*”;

- detaliile relevante cu privire la nodurile controlate sunt stocate în structuri de date care permit manipularea facilă a informațiilor, inclusiv pentru transmitere sub forma unor fișiere *plaintext* (XML;YAML);

- informațiile sunt livrate asincron, sub forma unor fișiere text XML care sunt decodate iar informațiile extrase;

- produsul final va fi structurat sub forma unor module Perl livrabile prin intermediul framework-ului de comunicații (bazat pe Ansible);

Concluzii

- platforma trebuie sa se bazeze pe module software preexistente, testate, optimizate, actualizabile;

- ușurința in administrare (atât din partea utilizatorului cât și a administratorului de sistem) este vitala;

- lărgirea domeniului de aplicabilitate a soluțiilor existente la resurse eterogene poate creste ușurința de administrare si implementare a soluțiilor server/client pentru o gamă largă de dispozitive;

Universitatea
Ştefan cel Mare
Suceava

Vă mulţumesc!

Această lucrare a beneficiat de suport financiar prin proiectul "Performanta sustenabila in cercetarea doctorala si post doctorala - PERFORM", Contract nr. POSDRU/159/1.5/S/138963", proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013.