Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en Computació d'Altes Prestacions

Gestor d'entorns virtuals per a l'execució de tasques d'altes prestacions

Memòria del treball de recerca del «Màster en Computació d'Altes Prestacions», dut a terme per Pau Tallada Crespí, sota la direcció de Miquel Àngel Senar Rosell, presentada a l'Escola Tècnica Superior d'Enginyeria (Departament d'Arquitectura de Computadors i Sistemes Operatius)

Juliol de 2009

Treball de recerca

Màster en Computació d'Altes Prestacions

Curs 2008-09

Gestor d'entorns virtuals per a l'execució de tasques d'altes prestacions

Dut a terme per Pau Tallada Crespí a l'Escola Tècnica Superior d'Enginyeria (ETSE) en el Departament d'Arquitectura de Computadors i Sistemes Operatius.

Director:
Miquel Àngel Senar Rosell

Autor:
Pau Tallada Crespí

A la meva família que tant estim i que tant m'estima, i als meus amics i companys amb els qui he compartit aquesta experiència. A tots ells, per tot el que m'han aportat i del que els seré sempre deutor.

Abstract

As the technology evolves the computational power increases. Past goals, which were deemed too difficult to achieve, now become computationally solvable. Most applications that focus on that problems are complex; they need a lot of resources to attain good performance, and that imposes a distributed architecture. Following the research community trend, in this work we propose an architectural design for distributed environments based on resource virtualization, which enables efficient resource management. The experimentations held have been able to prove this architecture viability, along with, how cou ld the use of virtual machines enhance resource management.

Keywords: grid, cloud computing, virtual machines, virtualization, distributed computing, resource management.

Sinopsi

Amb l'evolució de la tecnologia les capacitats de còmput es van incrementant i problemes irresolubles del passat deixen de ser-ho amb els recursos actuals. La majoria d'aplicacions que s'enfronten a aquests problemes són complexes, ja que per aconseguir taxes elevades de rendiment es fa necessari utilitzar el major nombre de recursos possibles, i això les dota d'una arquitectura inherentment distribuïda. Seguint la tendència de la comunitat investigadora, en aquest treball de recerca es proposa una arquitectura per a entorns grids basada en la virtualització de recursos que possibilita la gestió eficient d'aquests recursos. L'experimentació duta a terme ha permès comprovar la viabilitat d'aquesta arquitectura i la millora en la gestió que la utilització de màquines virtuals proporciona.

Paraules clau: entorns grid, computació cloud, màquines virtuals, virtualització, computació distribuïda, gestió de recursos.

Resumen

Con la evolución de la tecnología, las capacidades de cómputo se incrementan y problemas irresolubles del pasado dejan de serlo con los recursos actuales. La mayoría de las aplicaciones que se enfrentan a estos problemas son complejas, ya que para conseguir un elevado rendimiento es necesario utilizar el mayor número posible de recursos, lo que requiere de una arquitectura distribuida. Siguiendo la tendencia de la comunidad investigadora, en este trabajo de investigación se propone una arquitectura para entornos grid basada en la virtualización de recursos que posibilita la gestión eficiente de estos recursos. La experimentación llevada a cabo ha permitido comprobar la viabilidad de esta arquitectura y la mejora en la gestión que supone el uso de máquinas virtuales.

Palabras clave: entornos grid, computación cloud, máquinas virtuales, virtualización, computación distribuida, gestión de recursos.

Índex de continguts

Índex de figures

Índex de taules

1. Introducció

En les darreres dècades, el nombre d'ordinadors no ha parat d'augmentar i, amb ell, també ho han fet els recursos computacionals que conjuntament proporcionen. Aquest increment de les capacitats de còmput fa que, amb el temps, els problemes irresolubles del passat esdevinguin ferms candidats a ser resolts amb els recursos actuals.

Per a resoldre aquests problemes cal aprofitar al màxim les capacitats computacionals de què es disposa, i per a aconseguir un aprofitament eficient cal superar tres reptes inherents a l'elevat nombre de recursos.

- **Paral·lelisme:** La millor manera d'aprofitar el gran nombre d'ordinadors de què es disposa és executant-hi aplicacions paral·leles que siguin capaces d'escalar i executar-se en un gran nombre de nodes de manera simultània.
- **Organització en grid:** Tots aquests recursos computacionals es troben dispersats geogràficament, i sota dominis administratius diferents, el que provoca que es perdi el control sobre la plataforma operativa de que disposen. Per a poder utilitzar-los cal un cert nivell de coordinació i cooperació entre els diferents responsables i administradors.
- **Heterogeneïtat:** Finalment, trobam la gran variabilitat en les prestacions i configuracions del maquinari que conforma aquests recursos. En altres paraules, no es controla ni l'arquitectura, ni el sistema operatiu ni el programari que hi pugui haver instal·lat en cada una de les màquines.

Tots tres reptes suposen una dificultat afegida a l'hora de planificar i executar treballs a sobre d'aquests recursos. El repte del paral·lelisme és un repte que han d'assumir els desenvolupadors de les aplicacions. Per altra banda, existeix una tecnologia que resol la problemàtica derivada de l'heterogeneïtat i la pèrdua de control de la plataforma operativa, que és la virtualització. L'ús de màquines virtuals a sobre d'aquests recursos de còmput permet aconseguir la independència de l'arquitectura, el sistema operatiu, l'emmagatzemament, la xarxa i, fins i tot, de les llibreries o altres requeriments de programari que fossin necessaris per a l'execució. Altrament, l'ús de la virtualització, a més de resoldre tota la problemàtica esmentada, també aporta noves possibilitats i característiques al sistema:

- **Millora de la seguretat:** L'abstracció que s'aconsegueix amb l'ús de les màquines virtuals permet aïllar el sistema hoste físic del sistema convidat que s'hi executa a sobre, habilitant un nivells de seguretat molt elevats. A més a més, el fet de separar completament el sistema hoste permet executar-hi aplicacions no fiables, ja que els seus efectes queden confinats a l'àmbit de la màquina virtual.
- **Homogeneïtat:** L'ús de la virtualització permet l'execució de qualsevol imatge de màquina virtual en qualsevol màquines hoste. Aquesta característica aporta molta flexibilitat en tant que el planificador pot tractar de manera homogènia tots els recursos computacionals, ja que tots ells estan igualment capacitats per a executar qualsevol tasca.
- **Facilitat de reconfiguració:** L'abstracció que ofereix la màquina virtual permet pausar o interrompre l'execució d'una tasca o, fins i tot, migrar-la a un altre hoste. Aquesta característica ofereix noves possibilitats de cara a millorar la fiabilitat, disponibilitat i mantenibilitat (RAS) del sistema, tot gestionant de manera adequada les diferents tasques que s'executen.

Ara per ara, la majoria dels progressos en aquesta àrea es duen a terme de manera un tant dispersa. Existeixen diverses solucions de virtualització i multitud de sistemes de cues de treballs i planificadors diferents. En general, totes aquestes solucions estan poc integrades, i algunes encara es troben en estadis inicials del seu desenvolupament, o encara manquen d'algunes funcionalitats importants.

Tot i així, podem trobar alguns articles [1][2][3] que fan referència a la problemàtica d'integrar totes aquestes tecnologies i que proposen diferents arquitectures que permeten resoldre-ho. En les arquitectures plantejades, als recursos físics tradicionals d'un entorn grid s'hi afegeixen un nou conjunt de recursos virtuals, provinents d'aplicar les tecnologies de virtualització a aquests entorns. Una gestió adequada d'aquests recursos virtuals permet incrementar la capacitat de còmput del grid, ja que es poden fer servir com a plataforma d'execució de les tasques i, per altra banda, també proporcionen un nou conjunt de funcionalitats que aporten nous avantatges i una millora en la qualitat del servei que s'ofereix als usuaris del grid.

1.1. Objectius generals

En un projecte [4] de recerca anterior d'aquest mateix departament, es proposa una d'aquestes arquitectures per a la gestió dels recursos físics i virtuals d'un entorn grid orientada a l'execució eficient de les tasques dels usuaris. En aquesta arquitectura, certs tipus de treballs utilitzen màquines virtuals com a plataforma d'execució per tal d'aprofitar-se'n dels beneficis esmentats anteriorment. La idea és que cada usuari pugui desplegar un cloud “personal”, format per instàncies de màquines virtuals que el mateix usuari haurà configurat i proporcionat, amb la finalitat d'executar-hi les seves tasques amb la major eficiència possible.

Els usuaris transmeten aquestes imatges al sistema i en defineixen les seves característiques en un fitxer annex. En aquesta definició constarà un identificador únic que permet discriminar les diferents màquines virtuals. A l'hora d'enviar els treballs, aquests seran associats a les imatges mitjançant aquest identificador, i el sistema s'encarregarà d'aixecar les instàncies necessàries per a formar l'entorn d'execució demanat i gestionar-ne la seva execució. L'objectiu d'aquest projecte de recerca és automatitzar algunes d'aquests tasques, sobretot aquelles que afecten al cicle de vida dels treballs dels usuaris i de les màquines virtuals que els sustenten:

- **Implantació de les imatges de les màquines virtuals:** Les imatges de les màquines virtuals que proporcionin els usuaris com a plataforma d'execució poden no complir els requeriments necessaris per a poder ser gestionades de manera adient per aquesta arquitectura. Serà necessari implementar un procediment d'implantació que permeti instal·lar o adaptar les característiques de la màquina virtual per a que sigui compatible amb els components d'aquesta arquitectura.
- **Construcció de clouds privats per a l'execució de les tasques:** Els treballs que enviïn els usuaris podran executar-se sobre un conjunt d'instàncies de màquines virtuals. Caldrà, doncs, que un subsistema de l'arquitectura s'encarregui de gestionar i controlar aquestes instàncies, de manera completament transparent a l'usuari.
- **Gestió i execució de les tasques dels usuaris:** No podem oblidar que el fi de tota aquesta arquitectura és aconseguir una execució de les tasques dels usuaris amb una utilització més eficient dels recursos disponibles al grid. En aquest cas, l'arquitectura haurà d'implementar els mecanismes necessaris per a assegurar que l'usuari obté els resultats dels seus treballs correctament.

- **Millores RAS:** Ja s'ha comentat al punt introductori com l'ús de les tecnologies associades a la virtualització proporciona noves possibilitats de cara a millorar la fiabilitat, la disponibilitat i la mantenibilitat dels recursos computacionals que s'utilitzen. Així doncs, un altre dels objectius que es plantegen és integrar part aquestes noves possibilitats per tal de millorar les característiques RAS del sistema.
- **Aturada de les màquines virtuals i reciclatge de les instàncies:** Una vegada acabada l'execució de les tasques dels usuaris, les instàncies de les màquines virtuals poden restar a l'espera de l'arribada de nous treballs, o es poden apagar per alliberar els seus recursos. Un altre dels objectius d'aquest projecte serà avaluar les tècniques i criteris per poder dur a terme aquesta aturada, de manera que no es desaprofitin recursos i es maximitzi el reciclatge de les instàncies.
- **Estudi dels efectes de la suspensió i migració:** Finalment, el darrer objectiu d'aquest projecte és l'estudi dels efectes de la suspensió i la possible migració posterior de les màquines virtuals sobre el rendiment i la utilització dels recursos en un entorn grid.

Com a nota final, comentar que la tasca de construcció de les imatges de les màquines virtuals es deixa sota la responsabilitat de l'usuari final per dos motius: El primer és que es troba fora de l'àmbit d'actuació d'aquest projecte de recerca que, per la seva durada limitada, no pot abastar tots els objectius que es podrien plantejar. En tot cas, aquest aspecte es deixa com una línia oberta al treball futur. El segon motiu és de caire més pragmàtic: els usuaris que envien els seus treballs per a ser executats són els que coneixen millor, i de primera mà, els requeriments de la seva aplicació i, per tant, són els que es troben en millor posició per a generar les imatges de les màquines virtuals sobre les que s'hauran d'executar els seus treballs.

1.2. Organització del document

Aquesta memòria es divideix en sis capítols principals, més dos capítols addicionals de referències i bibliografia. El resum dels continguts dels capítols restants és el següent:

2) **Principis teòrics i estat de l'art:** Es detallen les tecnologies i àrees de coneixement implicades en aquest projecte i que permeten la comprensió del seu contingut. A més a més s'exposa l'estat de l'art de les diferents tecnologies i la manera com influeixen en l'arquitectura i el desenvolupament de la recerca.

3) **Disseny de l'arquitectura:** Exposa l'arquitectura del sistema que es proposa per tal de resoldre els objectius plantejats. Entra en detall en aquells punts en els que s'ha centrat l'experimentació.

4) **Implementació i experimentació:** En aquest capítol s'explica el desenvolupament i l'experimentació que s'ha dut a terme, centrats en la implementació d'alguns components de l'arquitectura definida al capítol anterior. De manera complementària a les tasques d'implementació també es duu a terme un estudi sobre possibles alternatives i camins de futur desenvolupament.

5) **Conclusions i línies obertes:** Aquest capítol resumeix les principals conclusions que s'han extret a partir de la recerca duta a terme en aquest projecte, així com una valoració d'aquells aspectes que més han influït en el seu desenvolupament. També s'han agrupat algunes de les idees que no s'han pogut emprendre i desenvolupar com calia, amb l'esperança de poder dedicar-hi futurs esforços a continuar la recerca en aquestes direccions.

2. Principis teòrics i estat de l'art

En aquest projecte de recerca conflueixen eines i tecnologies de diferents àrees que cal integrar per assolir els objectius proposats. Cada una d'aquestes àrees és una peça del trencaclosques de funcionalitats que juntes conformen l'entorn tecnològic del projecte. A grans trets es poden identificar tres grans àrees: per una banda les eines i plataformes de virtualització, que serveixen de base per a construir, configurar i instanciar les màquines virtuals. En segon lloc, els gestors de cues, que permeten encuar, prioritzar i gestionar gran quantitat de treballs de la manera més eficient possible. Finalment, trobam l'àrea dels gestors de clouds, que duen a terme la difícil tasca de desplegar, configurar i, en definitiva, gestionar un conjunt de màquines per tal d'assegurar que funcionen de manera coherent i coordinada.

En aquest capítol s'expliquen amb detall les tres àrees just esmentades i es dóna una breu visió l'estat de l'art de cadascuna d'elles.

2.1. Virtualització

El primer processador amb doble nucli aparegué l'any 2005, els de quatre nuclis el 2007 i els de vuit nuclis ja s'esperen per enguany. La llei de Moore no dóna cap treva i empeny al mercat i a la indústria a treure'n el màxim profit de les creixents capacitats de còmput. Per altra banda, mentre l'entorn d'escriptori roman impassible a aquest creixement, el mercat dels servidors aguaita amb fam i delit tota aquesta capacitat de processament, ja que sí disposa d'una àrea que podria aprofitar tot aquest potencial: la virtualització.

En els darrers anys s'ha viscut una explosió en el nombre d'articles i documents que versen sobre les tecnologies i eines associades a la virtualització, i les noves oportunitats que comporta: consolidació de recursos, balanceig de càrrega, reducció de costos, estalvi energètic, o un aprovisionament més ràpid, entre molts altres. Per altra banda, tots nosaltres ja fa temps que utilitzam la virtualització en el nostre ús diari dels ordinadors. De fet, cap de nosaltres no seria gaire productiu sense la virtualització que ens ofereixen els sistemes operatius moderns. En qualsevol ordinador recent, el sistema operatiu ja virtualitza la majoria dels seus recursos, com la memòria, el disc o el processador. És aquesta virtualització dels recursos la que ens ha proporcionat les capacitats de multitasca i multiprogramació de que feim un ús constant en el nostre dia a dia.

La virtualització és un dels conceptes més antics i fonamentals en les ciències de la computació i la podem trobar arreu, des d'ordinadors, telèfons mòbils, agendes electròniques, o fins i tot a les targetes gràfiques. La virtualització implica substituir un recurs dedicat per una abstracció d'aquest recurs que *sembla* dedicat -i es comporta com a tal-, però que en realitat es comparteix i està gestionat externament per assolir una major eficiència. És cabdal aquesta aparença de dedicació exclusiva, ja que simplifica en gran mesura el seu ús i, com que l'exclusivitat només és aparent, no existeix el perill que un sol usuari monopolitzi el recurs.

Avui en dia, el concepte renovat de virtualització respon a la idea d'utilitzar una capa addicional de programari, el monitor de màquines virtuals o hipervisor, per virtualitzar un sistema computacional complet, i no tan sols alguns dels seus components (memòria, disc o processadors) com es venia fent. La virtualització d'un sistema computacional sencer no és cap idea nova, ans al contrari; els seus inicis es remunten a la sèrie S/360 d'IBM, en particular al model S/360-40 [5]. Aquest model fou adaptat específicament per a executar CP-40, un sistema operatiu sorgit d'un projecte de recerca amb suport per a executar múltiples instàncies d'altres sistemes operatius clients, en concret CMS (Cambridge Monitoring System). Del CP-40 se'n derivà el CP-67, que tengué un gran èxit comercial com a part del sistema CP-67/CMS, del qual sorgiria més tard la família VM. Tot i així, des de la mateixa IBM, sobretot des del seu departament de ventes, no es veié amb gaires bons ulls la direcció i els bons resultats de la recerca en l'àrea de virtualització, ja que permetia reduir notablement el maquinari necessari per donar servei a un nombre determinat d'usuaris de temps compartit; i no podem oblidar que en aquell temps el negoci principal d'IBM era, precisament, la venta de maquinari.

Dit això, sorgeix la pregunta de, si els sistemes operatius moderns ja virtualitzen la majoria del maquinari, perquè és necessari fer un pas més i virtualitzar el sistema computacional sencer? Els sistemes operatius actuals proporcionen a cada procés un determinat espai de memòria en exclusiva, però al mateix temps comparteixen el sistema de fitxers, el processador i la instància del sistema operatiu. En moltes situacions, aquest aïllament pot ser insuficient, ja que un procés pot arribar a ofegar als altres en l'accés als recursos, per molt bona que sigui la política de planificació. En un entorn amb el maquinari completament virtualitzat, cada màquina virtual disposa de la seva pròpia instància del sistema operatiu, de la seva memòria i sistemes de fitxers d'ús exclusiu, i la seva comunicació només serà possible mitjançant una xarxa virtual.

La virtualització completa del sistema s'aconsegueix mitjançant la reducció dels privilegis dels sistemes operatius clients, i l'establiment d'una capa d'abstracció superior que els gestiona. Els privilegis en un ordinador s'estructuren com un conjunt d'anells concèntrics, on les aplicacions que s'executen a l'anell interior disposen de privilegis totals, mentre que les que ho fan als anells exteriors veuen reduïdes les seves capacitats i privilegis. Els sistemes operatius tradicionals s'executen a l'anell 0 (el més intern) i, per tant, disposen de privilegis complets sobre el sistema. En un entorn virtualitzat, els sistemes operatius clients s'executen a l'anell de privilegis 1, mentre que a l'anell 0 s'executa aquesta nova capa d'abstracció. A aquesta capa se l'anomena monitor de màquines virtuals o hipervisor, i la seva funció principal és arbitrar l'accés al maquinari subjacent per part dels sistemes operatius clients.

Als inicis de la virtualització, amb l'IBM S/370, el mecanisme que permetia a l'hipervisor controlar els sistemes operatius clients era molt robust. Tota instrucció que s'intentava executar amb un nivell insuficient de privilegis causava una interrupció, que era captat per l'hipervisor, i aquest *emulava* el comportament de la instrucció de manera que la resta de màquines virtuals no es veiessin afectades. En altres paraules, la virtualització de l'IBM S/370 estava assistida pel maquinari, ja que era l'encarregat de cedir el control a l'hipervisor quan detectava que una màquina virtual intentava accedir a un recurs privilegiat. La detecció de les interrupcions i el posterior traspàs de control no és una operació senzilla, sinó que sol tenir un cost molt elevat. D'aquí que per tal de maximitzar el rendiment, els desenvolupadors d'IBM intentaren minimitzar el nombre de crides privilegiades, així com el temps que es tardava en *emular* aquestes crides.

Tot i que la virtualització assistida per maquinari ja estava plenament implementada i en ús als anys 70, no arribà als ordinadors personals fins fa pocs anys [6]. Fins aleshores, era necessari recórrer a solucions basades en programari per tal d'aconseguir una virtualització completa del sistema. En els punts següents s'expliquen algunes de les tècniques emprades per tal d'assolir aquesta virtualització.

2.1.1. Virtualització assistida per programari

En l'arquitectura x86, absolutament majoritària en els entorns d'escriptori, fins fa ben poc no era possible la virtualització assistida pel maquinari. El principal problema fou que el microprocessadors i l'arquitectura d'instruccions (ISA) no foren dissenyades tenint en compte

els requeriments de la virtualització. En concret, els primers models de microprocessadors x86 foren dissenyats per ser usats en calculadores, i la virtualització no estava a la llista de funcionalitats que s'esperaven. D'aquesta manera, segons el disseny original que encara hereten els processadors actuals, el conjunt d'instruccions x86 no llança una interrupció en tots els casos en els que el control hauria de ser cedit a l'hipervisor, sinó que en alguns casos simplement ignora la instrucció. El resultat és que el sistema operatiu client pensa que ha dut a terme una acció que en realitat no s'ha executat, i l'hipervisor ni tan sols ha pogut detectar aquest succés.

Segons [7], el conjunt d'instruccions de l'arquitectura x86 conté fins a 17 d'aquestes instruccions que creen problemes de cara a la virtualització, doncs permeten llegir i modificar estructures sensibles, com taules de descriptors o els registres d'estat del sistema. La conclusió a tot això és que l'arquitectura x86 no pot ser virtualitzada de la mateixa manera que ho foren els antics *mainframes*. Altrament, les arquitectures PowerPC i Alpha sí són prou consistents com per poder ser virtualitzades a la manera clàssica.

Traducció binària

Aquestes errades de disseny en l'arquitectura d'instruccions de la plataforma x86 no impediren que a l'any 1999 VMware llancés una de les primeres solucions de virtualització completa sense basar-se en l'ajuda del maquinari. La tècnica utilitzada per assolir aquest objectiu s'anomena traducció binària, i és molt més lleugera i eficient que les implementacions utilitzades als processadors Itanium o Transmeta, ja que no ha de traduir entre dues ISA diferents, sinó que al treballar només amb x86 molts cops és suficient amb copiar la instrucció original.

Amb aquesta tècnica, VMware tradueix el codi binari del nucli del sistema operatiu client al vol just abans d'executar-se. En la majoria dels casos, el codi traduït del nucli serà una còpia exacta; només en aquells casos en els que el nucli hagi d'executar una instrucció privilegiada, el traductor la substituirà per una secció de codi una mica més llarga que durà a terme la mateixa operació però de manera més segura. Per exemple, en aquells casos on el sistema operatiu client desitgi prendre el control d'algun recurs de maquinari, el traductor binari substituirà aquella secció de codi amb una altra que manipularà el maquinari *virtual*.

Per altra banda, el codi de les aplicacions d'usuari no es tradueix, ja que no suposa cap amenaça, i s'executa directament com si fos codi nadiu.

En altres paraules, la traducció binària no és res més que analitzar el codi que el nucli d'un sistema operatiu client vol executar en un moment determinat i anar substituint aquelles instruccions privilegiades per una secció de codi segura, on la seguretat s'avalua respecte a l'hipervisor i a la resta de màquines virtuals. A més, el codi substituït es desa en una memòria cau per tal de reutilitzar-lo més endavant, de manera que en cas de presència de bucles, la traducció només es fa un cop.

Per altra banda, hi ha certes situacions en les que es fa evident la complexitat de l'ús d'aquesta tècnica. Per exemple, en el cas dels salts a una altra ubicació de memòria, el traductor binari no pot copiar l'adreça de destí, ja que si ha inserit codi en alguna posició intermèdia el destí correcte es pot trobar desplaçat respecte a l'adreça indicada. En aquests casos cal corregir l'adreça de destí per tal que el flux de control no quedi alterat per aquest procés de traducció.

Paravirtualització

Una altra de les tècniques per assolir la virtualització completa sense el suport del maquinari és la paravirtualització. La paravirtualització és una tècnica molt semblant a la traducció binària: mentre que el procés de traducció es duu a terme al vol sobre el codi binari que està a punt de ser executat, la paravirtualització requereix que els canvis es duguin a terme sobre el codi font. És a dir, la paravirtualització requereix modificacions al sistema operatiu client, de manera que pugui ser conscient que s'està executant en un entorn virtualitzat i pugui adaptar el seu comportament. És evident que la flexibilitat que es pot obtenir modificant el codi font és superior a la traducció binària al vol. A més a més, la paravirtualització permet eliminar moltes interrupcions innecessàries, fins i tot més que amb la traducció binària.

En un sistema paravirtualitzat, l'hipervisor ofereix una interfície addicional de crides al sistema, o *hipercrides* per dur a terme aquelles operacions crítiques com són la gestió de la memòria o el control de les interrupcions. Aquestes hipercrides permeten optimitzar el traspàs de control entre l'hipervisor i el sistema operatiu paravirtualitzat, ja que aquest darrer, al ser

conscient de trobar-se en un entorn virtualitzat, es comporta de manera «civilitzada» i recorre a les hipercrides cada cop que li cal dur a terme una operació privilegiada.

Com a petit resum, la paravirtualització és un concepte molt interessant, ja que permet eliminar completament la sobrecàrrega del procés de traducció binària, a la vegada que també redueix el nombre d'interrupcions que l'hipervisor ha d'interceptar per assegurar el correcte aïllament de les diferents màquines virtuals.

2.1.2. Virtualització assistida per maquinari

Per una banda, la paravirtualització és una molt bona tècnica per assolir la virtualització completa d'un sistema operatiu client, però exigeix modificacions en el seu codi font. Emperò, no sempre es disposa del codi font ni del permís per a modificar-lo, cas en el que s'haurà de recórrer a altres tècniques com la traducció binària. Per altra banda, la traducció binària també permet aconseguir l'objectiu de la virtualització completa, tot i que amb un major cost de processament. A més a més, hi ha certs aspectes on la traducció binària té molt poc marge de maniobra i es veu severament limitada, com en el cas de l'emulació de les crides a sistema, les operacions d'E/S, interrupcions o transferències directes a memòria (DMA), la gestió de la memòria, o els codis extremadament complexos (polimorfisme, codi mutant, fluxos de control indirectes, entre d'altres). En aquests casos, l'assistència del maquinari pot resultar útil.

Com ja s'ha comentat, la virtualització assistida pel maquinari ja es va implementar els anys 70 a la sèrie S/370 d'IBM, però l'arquitectura x86 no va estar preparada per a la virtualització clàssica fins a principis del segle XXI. La idea principal que cal que quedi clara és que la virtualització assistida pel maquinari no és una versió millorada de la traducció binària o de la paravirtualització. És a dir, malgrat el suport del maquinari, l'objectiu no és augmentar el rendiment de la virtualització, sinó solucionar els errors de disseny de l'arquitectura x86 que han impedit la seva virtualització clàssica. La virtualització assistida pel maquinari en x86 es basa en la filosofia que ja imperava als S/370, només que un tant millorada; és a dir, en capturar totes aquells instruccions privilegiades susceptibles de rompre l'aïllament de les diferents màquines virtuals i emular-ne la seva execució, de manera que els sistemes operatius clients fossin aliens a la presència d'un hipervisor i es comportessin de manera idèntica a com ho farien si s'executessin nativament.

El que han fet els diferents fabricants [8], en particular Intel i AMD, ha estat ampliar el conjunt d'instruccions amb algunes de noves que permetin la gestió del traspàs de control entre l'hipervisor i les diferents màquines virtuals, i afegir un nou nivell d'execució destinat a contenir l'hipervisor. En el cas d'Intel, aquesta tecnologia s'ha anomenat Intel-VT, amb el nom en codi Vanderpool, mentre que AMD l'ha batejat com a AMD-V, amb el nom en codi Pacífica.

Un gran avantatge d'aquesta tècnica és que els sistemes operatius clients s'executen al nivell de privilegis estàndard (l'anell 0), mentre que l'hipervisor s'executa en un anell amb un nivell encara major de privilegis: l'anell -1. D'aquesta manera, les crides a sistema ja no resulten en una intervenció directa de l'hipervisor: mentre aquestes crides a sistema no impliquin instruccions privilegiades, l'hipervisor se'n mantendrà al marge. Això suposa un avantatge molt important per a la virtualització assistida per maquinari.

La problemàtica inherent en aquesta tècnica és que cada transició de la màquina virtual a l'hipervisor, i a la inversa, suposa un consum fix i elevat de cicles de processador. El nombre concret d'aquests cicles de sobrecàrrega depèn de la implementació interna del processador i, depenent de l'operació, pot arribar a ser de l'ordre dels milers de cicles. Així doncs, el traspàs de control és una operació molt costosa. La manera en que els fabricants de processadors actuals s'han enfrontat a aquest problema ha estat reduir el nombre de cicles que tarda aquesta transferència de control, a la vegada que es redueix el nombre de situacions en les que cal dur a terme aquesta transferència.

En la majoria dels casos, la paravirtualització i la traducció binària assoleixen millor rendiment que la virtualització assistida pel maquinari i, possiblement, l'escenari òptim sigui un hipervisor que sigui capaç de combinar el millor de tots les tècniques disponibles, en el que es coneix com a virtualització híbrida [9].

2.1.3. Programari de virtualització

De les primeres solucions de virtualització orientades a l'entorn d'escriptori, la majoria sortiren de projectes de recerca dins universitats, mentre que d'altres sorgiren com a fruit de la dedicació personal d'alguns entusiastes. És per això que la majoria eren simples proves de concepte, amb un caire més experimental que no pas dirigides a l'usuari final. Als anys 90, de les primeres eines de virtualització que es posaren a disposició del gran públic en podem

destacar dues. Per una banda Basilisk II [E1], un emulador per a plataformes Macintosh basades en el processador Motorola 68000 i que permet l'execució de totes les versions de MacOS fins a la 8.1. Per l'altra banda trobam QEMU [E2], possiblement un dels hipervisors més versàtils que existeixen actualment ja que permet emular fins a vuit arquitectures de maquina diferents, com Intel x86, ARM, SPARC, MIPS o PowerPC, entre d'altres. De fet, QEMU ha esdevingut amb el temps el pare d'alguna de les solucions de virtualització més potents del mercat actual. Ambdós productes fan ús de la tècnica de traducció binària, però el fet d'haver de traduir d'una arquitectura a una altra completament diferent fa que el rendiment que aconsegueixen sigui molt pobre, limitant així les seves possibles aplicacions comercials.

En els darrers anys, sobretot degut a l'explosió en les capacitats de còmput dels ordinadors moderns, s'han produït avenços enormes en el rendiment i en les expectatives posades en les tecnologies de virtualització que han permès el seu desplegament a nivell comercial. Aquesta entrada a l'àmbit comercial ha provocat també una explosió molt gran en el nombre de funcionalitats proporcionades per aquestes tecnologies, així com certs canvis en la mateixa arquitectura del maquinari per a poder aprofitar millor les seves capacitats. En aquest darrer aspecte ja s'ha fet esment a tot un conjunt d'instruccions que els fabricants de microprocessadors han afegit als seus xips per donar suport explícit a la virtualització [E6] [E7].

De les eines mes representatives que es troben disponibles caldria ressaltar-ne tres. En primer lloc VMware [E3], una de les eines de virtualització amb més projecció comercial del moment. VMware emula completament la plataforma de maquinari subjacent i, mitjançant tècniques de traducció binària i l'aprofitament de les extensions pròpies dels microprocessadors, és capaç d'assolir unes taxes de rendiment molt bones. En segon lloc tenim Xen [10][E4], una alternativa de codi obert que ha tengut molt bona acceptació, sobretot entre la comunitat del programari lliure. Tot i poder aprofitar les extensions de virtualització dels processadors moderns, una de les claus del seu èxit han estat les modificacions sobre el nucli Linux per a permetre'n la paravirtualització [E8]. Tot i així, la complexitat d'aquestes modificacions i la poca col·laboració dels seus desenvolupadors amb els responsables del nucli Linux han impedit que aquests canvis s'integressin a la branca oficial. De fet, aquesta manca d'integració amb la branca oficial ha propiciat que altres alternatives prenguin força. Una de les alternatives que ha agafat més volada és KVM [E5], una solució també de programari lliure que, al contrari que Xen, parteix amb la benedicció

dels desenvolupadors del nucli Linux, ja que està integrat a la branca oficial des de versió 2.6.20, fa més de dos anys. El seus principals avantatges són dos: per una banda, el fet de requerir la presència de les extensions de virtualització als processadors ha permès simplificar molt el codi, prescindint de les complexes tècniques de reescriptura binària, i facilitant la seva inclusió a la branca oficial del nucli Linux. I per altra banda, el fet d'estar inclòs de sèrie al nucli Linux l'ha dotat d'una base d'usuaris molt gran i en contínua expansió i, amb el desplegament imparable de les extensions de virtualització, és probable que en poc temps es situï com una de les solucions més àmpliament usades en els entorns Linux.

2.2. Entorns grid

El terme *grid* fou introduït per primer cop l'any 1998 en un llibre [11] escrit per Ian Foster i Carl Kesselman. El llançament d'aquest llibre fou el detonant que impulsà la recerca en una nova àrea que tot just definien: la computació en entorns grid. Les idees i definicions originals establien una analogia entre els entorns grid computacionals i la infraestructura de la xarxa elèctrica. Aquesta primera definició que donaren fou:

> *Un grid computacional és una infraestructura de maquinari i programari que proporciona un accés confiable, consistent, omnipresent i a baix cost a unes capacitats molt elevades de còmput.*

De fet, la idea de l'accés sota demanda a recursos de computació ja havia estat suggerit, potser prematurament, el 1969 per Len Kleinrock:

> *Amb tota probabilitat veurem l'expansió de «serveis de computació» que, tal com els serveis d'electricitat i telefonia actuals, proporcionaran capacitats de còmput a llars i oficines arreu del país*

L'any 2000 el mateix Ian Foster en un altre article, adapta la definició de grid per a incloure-hi els aspectes polítics i socials, establint que la computació en entorns grid implica «la compartició coordinada dels recursos i la resolució de problemes en un entorn dinàmic format per organitzacions virtuals multiinstitucionals». També féu èmfasi en la importància d'establir protocols estandarditzats per a permetre la interoperabilitat entre les diferents infraestructures.

En un article [12] de l'any 2002, el mateix autor suggereix un conjunt de tres regles per les quals defineix un sistema grid com aquell que:

1. *coordina recursos que no estan gestionats de manera centralitzada...*

   Els recursos que integren un grid provenen de diferents dominis administratius, i solen estar distribuïts geogràficament. La infraestructura del grid ha de proporcionar els serveis necessaris per a gestionar els aspectes de seguretat, polítiques d'ús, pagament i membresia. En els altres casos estaríem parlant d'un sistema de gestió local.

2. *mitjançant protocol i interfícies oberts, estàndards i de propòsit general...*

   Un grid està compost per múltiples protocol i interfícies de propòsit general que s'encarreguen de serveis tan fonamentals com l'autenticació, l'autorització, el descobriment de recursos i la gestió dels accessos. En tots aquests casos, els protocols han de ser oberts i estàndards. En cas contrari estaríem treballant amb una sistema dirigit per una aplicació o protocol de propòsit específic.

3. *per proporcionar diversos graus de qualitat de servei*

   L'objectiu del grid és utilitzar els seus recursos constituents de manera coordinada per a per a proporcionar diversos graus de qualitat de servei, com temps de resposta, productivitat, disponibilitat, fiabilitat o seguretat, enfocats a donar un servei adequats a les complexes necessitats dels seus usuaris, de manera que la utilitzat del conjunt de recursos del grid sigui major que la suma de les seves parts.

Altres autors [13] proposen definicions diferents del concepte de grid, com les cinc grans idees proposades per Grid Café [E14], o la visió proporcionada per Heinz Stockinger [14], molt més completa. Per altra banda, el concepte de grid de computació, com a concepte simètric a la xarxa elèctrica, encara no és real; ara per ara, tan sols existeixen multitud de petits grids isolats i no integrats. Tot i així, aquests grids aïllats ja proporcionen un servei molt valuós a alguns projectes científics, que sense aquesta infraestructura no serien factibles. Els grids permeten als biòlegs estudiar el plegament de les proteïnes per a la detecció de nous medicaments, als geòlegs processar els nivells d'ozó a l'atmosfera, als físics d'altes energies simular i analitzar col·lisions de partícules, als artistes generar complexes animacions per a la indústria cinematogràfica, o als sociòlegs fer recerca sobre la vida social de les abelles.

En resum, l'objectiu dels entorns grids es proporcionar els recursos necessaris per a executar tasques, la majoria de les quals pertanyen a projectes científics que, sense l'ajut d'aquesta infraestructura, no es podrien dur a terme. Per altra banda, els entorns grids permeten que la informació amb que es treballa estigui distribuïda per tot el món i, d'aquesta manera, els diversos equips científics que participen en un projecte poden treballar amb comoditat dels dels seus propis laboratoris.

2.2.1. Middleware d'entorns grid

Els grids permeten compartir, intercanviar, descobrir, seleccionar i agregar tot un conjunt de recursos heterogenis geogràficament distribuïts, tals com ordinadors, bases de dades, dispositius de visualització o instruments científics. D'aquesta manera s'estan convertint en una plataforma de nova generació i infraestructura global sobre la qual resoldre reptes científics de gran escala. Però gestionar tots aquests recursos i infraestructura no és una tasca gens senzilla; s'han dedicat grans esforços en el disseny i implementació de programari middleware amb l'objectiu de fer-ho possible.

Algunes d'aquestes solucions de programari s'han desplegat amb èxit i ja és possible construir grids que abasten més enllà dels dominis d'una xarxa local. A la figura 1 es pot observar l'arquitectura típica d'un grid, formada per quatre capes:

- **Fabric:** Aquesta capa està composta per tots els recursos distribuïts, com ordinadors, xarxes, dispositius d'emmagatzemament i instruments científics. Els recursos computacionals poden correspondre a diverses arquitectures i organitzacions (clusters, supercomputadors, ordinadors personals, etc.) i poden executar diferents sistemes operatius. Els instruments científics poden proporcionar dades en temps real o anar-les desant en una base de dades
- **Core Grid Middleware:** Proporciona serveis com la gestió remota dels processos, reserva de recursos, accés a l'emmagatzemament, descobriment dels sistemes d'informació, seguretat i qualitat de servei. Mitjançant aquesta capa és possible abstreure la complexitat i l'heterogeneïtat dels components de la capa *fabric*, oferint un accés uniforme i consistent als recursos distribuïts.

- **User-level Grid Middleware:** Proporciona un nivell superior d'abstracció i nous serveis com entorns de desenvolupament, eines de programació i planificadors de recursos, basant-se en les interfícies proporcionades per la capa inferior.
- **Grid applications and portals:** Les aplicacions es solen desenvolupar emprant llenguatges preparats per a entorns grids, com HPC++ o MPI, i solen requerir de grans capacitats de còmput, bases de dades distribuïdes, accés a instruments científics o una combinació dels mateixos. Els portals grid ofereixen aplicacions web on els usuaris poden trametre i consultar els seus treballs que s'executen distribuïts pels recursos del grid.

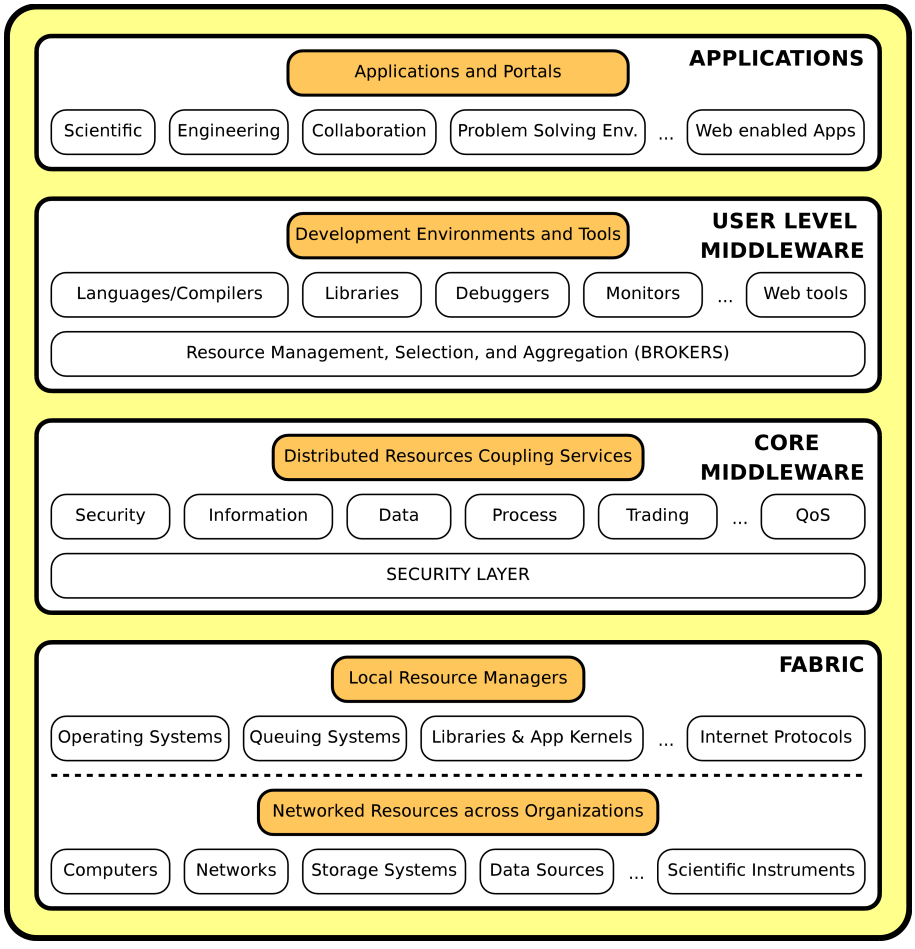

Figura 1: Estructura i components típics d'un grid

Globus

Com a bon exemple de middleware per a gestionar grids trobam Globus [E15], projecte de programari lliure impulsat per la «Globus Alliance», que agrupa organitzacions com

l'Argonne National Laboratory o l'Information Sciences Institute, entre d'altres. A la figura 2 es pot observar l'especialització de l'arquitectura de Globus [15], basada en l'esquema genèric de la figura 1.

Globus agrupa els serveis proporcionats en tres grups principals, que funcionen tots sobre una capa de seguretat. A continuació es descriuen amb una mica més de detall el funcionament i responsabilitats de cada una d'aquestes capes:

- **GSI Security Layer:** Aquesta capa permet autentificar els usuaris del grid així com assegurar la confidencialitat i integritat de les comunicacions. Està basada en les tecnologies de clau pública (PKI), SSL i certificats X.509. Entre les funcionalitats que implementa aquesta capa trobam el punt d'entrada únic (*single sign-on*), autenticació de recursos i usuaris mitjançant certificats, xifratge de dades, etc.
- **Resource Management:** El grup de gestió de recursos permet la reserva dels recursos en l'enviament de treballs, el seguiment dels mateixos i la recollida dels resultats. Globus disposa de dos components en aquest grup que implementen aquestes funcionalitats:
  - **Globus Resource Allocation Manager (GRAM):** Proporciona les funcionalitats d'enviament i monitorització remota de treballs, així com la posterior recollida dels resultats. A l'hora d'executar el treball, aquest component cedeix el control a un gestor de cues local, com *Portable Batch System* (PBS) o *Load Sharing Facility* (LSF), entre d'altres.
  - **Globus Access to Secondary Storage (GASS):** Es un sistema de fitxers especial que permet a les aplicacions dur a terme la precàrrega de dades, així com llegir i escriure sobre fitxers remots.. També gestiona l'entrada i la sortida estàndard dels treballs que s'executen.
- **Information Services:** Aquest grup de serveis és l'encarregat de descobrir, emmagatzemar, organitzar i distribuir tot un conjunt de propietats o atributs (tant estàtics com dinàmics) dels recursos associats a un grid. El component concret que s'encarrega d'aquesta funció s'anomena Monitoring and Discovering Service (MDS). Aquest component s'estructura internament com un conjunt de capes, on les inferiors són les responsables d'obtenir i generar aquests atributs, mentre que els capes

superiors són les encarregades de recollir, organitzar i emmagatzemar tota aquesta informació, a la vegada que arbitren la seva consulta.

- **Data Management:** Proporciona eines i biblioteques per a la transmissió, emmagatzematge i gestió de grans volums de dades, utilitzades molt sovint per algunes aplicacions científiques. Globus divideix els serveis d'aquest grup en dos components:
  - **GridFTP:** És una extensió al protocol estàndard FTP que proporciona transferència segura, confiable i eficient de dades en entorns grid.
  - **Replica Location and Management:** Proporciona replicació i distribució de múltiples còpies per a un mateix fitxer o font de dades.

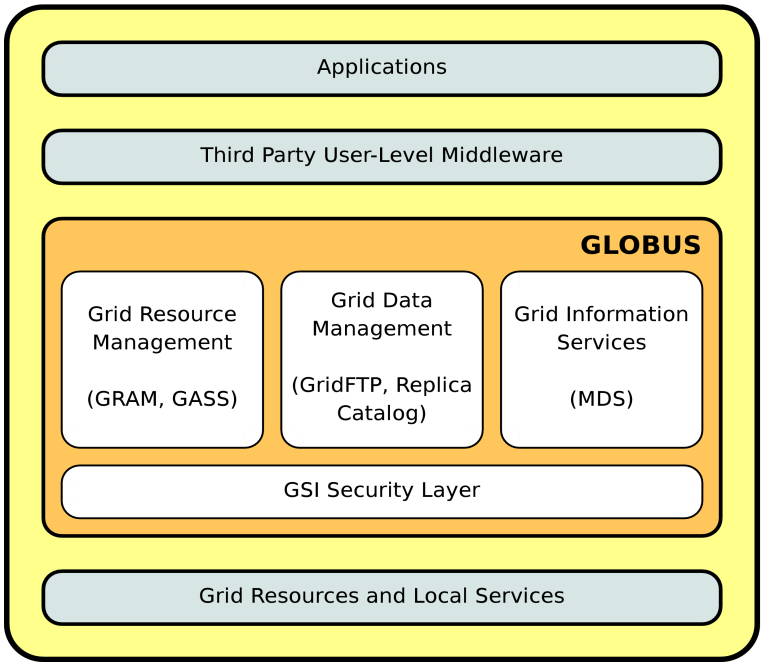

Figura 2: Arquitectura de Globus

2.3. Gestors de cues

Ja s'ha comentat que l'objectiu dels grids és proporcionar un entorn adequat per a que les aplicacions que treballen amb grans volums de dades o que requereixen d'una elevada potència de còmput puguin executar les seves tasques. Tot i que els grids proporcionen la infraestructura, l'execució final d'aquests treballs és responsabilitat dels gestos de cues locals,

que són els encarregats de distribuir aquests treballs entre tots els recursos disponibles d'acord amb les polítiques que marquin els administradors del sistema.

L'objectiu principal d'aquest projecte de recerca és permetre a l'usuari executar les seves tasques de la manera més eficient possible, utilitzant per a aquest fi les màquines virtuals instanciades en un entorn grid. Cal, doncs, també un gestor de cues per gestionar els treballs que l'usuari envia i monitoritzar la seva execució dins les màquines virtuals.

Existeixen diverses eines al mercat amb aquest propòsit, de les quals en podem destacar les tres següents:

- **OpenPBS [E9]:** Un dels primers gestors de cues, desenvolupat per la NASA als anys 90 per a plataformes UNIX. A l'any 2001 la branca alliberada com a programari lliure es deixà de mantenir i, actualment, el desenvolupament només continua en mans de l'empresa Altair en forma d'una versió propietària. Aquesta versió propietària, anomenada PBSpro [E16], és compatible amb Globus des del 1999 i es composa de tres serveis o dimonis. Un dimoni per gestionar cada clúster, un dimoni planificador (es pot incloure un planificador extern de manera opcional) i un dimoni MOM (*Machine Oriented Mini-Server*) que s'executa a cada node d'execució que s'encarrega de l'execució i el control dels treballs. Ofereix tolerància a fallades, permet pausar i reprendre les tasques, *checkpoints*, suport per a SMP i integra un frontal grid cap a Globus.
- **Sun Grid Engine (SGE) [E10]:** Conegut anteriorment com a CODINE (*Computing in DIstributed Networked Environments*) o GRD (*Global Resource Director*) és un altre gestor local de cues amb llicència lliure desenvolupat i mantingut per Sun Microsystems. Sun també proveeix una versió propietària basada en SGE anomenada *N1 Grid Engine* (N1GE). Les funcions d'SGE són les pròpies de qualsevol gestor de cues: encuar, planificar, gestionar i controlar l'execució remota i distribuïda d'un gran nombre de tasques individuals, paral·leles o interactives que els usuaris envien, així com també de la planificació i reservar dels recursos distribuïts (processadors, memòria, emmagatzemament, etc.)
- **Condor [E11]:** És una plataforma de programari lliure orientada a la gestió i paral·lelització de tasques amb alts volums de còmput, desenvolupada a la universitat de Wisconsin-Madison, sota la llicència lliure Apache 2.0. Condor és capaç

d'executar-se sobre diversos sistemes operatius, incloent Linux, Unix, Mac OSX, FreeBSD i Windows i es pot instal·lar tant en màquines dedicades (clústers en rack) com també per aprofitar els cicles inactius de les màquines d'escriptori.

Condor implementa alguns components per integrar-se en entorns grids, com Condor-G, que permet reenviar els treballs cap a un planificador extern. Per altra banda, Condor és un dels gestors de cues compatibles amb el component GRAM de Globus. Com a exemple del seu bon renom, comentar que fou el gestor de cues emprat per a distribuir els treballs a l'hora de generar el primer esbós del genoma humà. Finalment comentar que Condor implementa funcionalitats específiques per a executar treballs com a màquines virtuals, a través de diverses plataformes de virtualització (com Xen, VMware o KVM).

2.4. Gestió de màquines virtuals i entorns cloud

En els entorns grid, és necessària una adequada gestió i administració dels recursos de cara a executar de manera eficient els treballs dels usuaris. Si a aquest entorn hi afegim les capacitats de virtualització que ofereixen els ordinadors actuals, podem transformar aquest grid en una plataforma capaç d'executar gran quantitat de màquines virtuals. Aquestes màquines virtuals es poden gestionar com si fossin treballs del grid, i es poden organitzar en forma de cloud [16].

L'origen dels clouds, o del terme *Cloud Computing* es remunta a l'any 2006, quan Amazon començà a oferir màquines virtuals a 0,10 $/hora mitjançant una API molt senzilla. La posada en funcionament d'aquest servei va suposar l'acostament del *cloud computing* al gran públic, ja que les seves notables prestacions, ofertes a uns preus realment irrisoris, feren palès al gran públic que es trobaven davant d'un concepte revolucionari. El nom comercial d'aquest servei és Amazon Elastic Compute Cloud (EC2) [E12], i s'ofereix com un servei de lloguer de màquines virtuals, constituïdes a mode de cloud privat, completament dinàmic i personalitzable al gust de l'usuari. L'èxit i la popularitat que assolí aquest servei impulsaren a la resta de fabricants a etiquetar les seves solucions com a «Cloud». D'aquesta manera, la industria accepta actualment tres formes de clouds: Infraestructura com A Servei (IaaS), com Amazon EC2; Plataforma com A Servei (PaaS), que ofereix un entorn prefixat per desplegar aplicacions sobre un proveïdor, com Google App Engine [E17]; i Programari com A Servei

(SaaS), que no és res més que programari només accessible per Internet, com Google Docs [E30]. La originalitat del que generalment s'anomena Cloud, es correspon a aquest primer tipus, la Infraestructura com A Servei (IaaS), i, de fet, gràcies a la seva generalitat es sol utilitzar com a base per a les dues formes restants. La seva contribució més important és haver aconseguit proporcionar capacitat de processament com si es tractés de qualsevol altre bé de subministrament, com l'electricitat o l'aigua, en un concepte anomenat *Utility Computing*.

En els entorns grid, els usuaris comparteixen i competeixen per l'ús dels diferents recursos que el formen (processadors, memòria, emmagatzemament) amb la finalitat de poder-hi executar els seus treballs. En un entorn cloud, gràcies a la virtualització dels serveis s'assoleix un nivell d'abstracció major, pel qual els usuaris ja no han de compartir els recursos. En un entorn cloud virtualitzat, els recursos es segmenten i s'assignen a les diferents màquines virtuals. Al mateix temps els usuaris sol·liciten un cert nombre de recursos, en forma de màquines virtuals, amb unes característiques determinades de memòria i capacitat de còmput corresponents a les necessitats de les seves tasques, i en les que es duria a terme la seva execució. Aquest conjunt de màquines virtuals assignades a un usuari per a l'execució de les seves tasques formaria un cloud personal o privat de l'usuari. D'aquesta manera, les tasques dels usuaris s'executen en un entorn aparentment dedicat i amb uns recursos destinats en exclusiva.

És per això que el component més important per al desplegament d'aquests entorns cloud és el gestor de màquines virtuals, que ha de ser capaç d'oferir una visió integrada de tots els recursos de la infraestructura cloud, independentment de la tecnologia de virtualització que s'hagi utilitzat. A més a més, ha de permetre l'ús de diferents polítiques d'assignació de recursos per tal de poder assolir objectius diferents (com una reducció del temps de resposta o una major ocupació dels recursos, entre d'altres). Finalment, aquesta funcionalitat ha de ser accessible de forma segura des de l'exterior mitjançant senzilles interfícies.

En els darrers temps s'han desenvolupat diversos projectes amb l'objectiu de transformar una infraestructura existent en un cloud IaaS, i que inclouen tant un gestor de màquines virtuals com interfícies d'accés tipus cloud. Globus Nimbus [E31] es un servei per instanciar màquines virtuals que implementa una interfície pròpia així com un servei web compatible amb la interfície oferta per Amazon EC2. Nimbus inclou infraestructura addicional per construir clouds científics, destinats a instanciar plataformes virtuals per computació. Emperò,

el projecte cloud de programari lliure amb més acceptació possiblement sigui Eucalyptus [E13], que implementa de forma pràcticament íntegra les interfícies d'Amazon EC2 i Simple Storage Service (S3). El fet d'oferir una interfície completament compatible amb Amazon EC2 respon a l'objectiu de facilitar al màxim que els usuaris puguin migrar al seu servei sense haver de fer gaires canvis en l'estructura dels seus sistemes.

Per altra banda, tot i que aquests projectes representin una solució funcional respecte a les interfícies cloud i mecanismes de seguretat, encara manquen d'algunes de les característiques típiques que requereix un gestor de màquines virtuals. Per exemple, una de les mancances més rellevants és el fet que l'assignació de recursos es fa de manera estàtica i seguint algorismes molt senzills com round-robin. Per tant, a l'actualitat encara no existeix una pila completa de programari que permeti implementar correctament una solució cloud, en la que s'asseguri la gestió eficient dels recursos locals i la implementació d'interfícies remotes senzilles.

3. Disseny de l'arquitectura

Una vegada desplegat l'àmbit tecnològic en el que es centra aquest projecte, en aquest capítol es descriuen de manera més detallada els objectius que es pretenen assolir, així com el disseny d'una l'arquitectura que és capaç d'assolir-los.

En el primer apartat es descriu l'abast del projecte fent referència als treballs anteriors en els que es fonamenta. A continuació s'aprofundeix una mica en la problemàtica que es pretén resoldre, els diferents enfocaments i s'analitzaran els requeriments necessaris per emprendre la implementació.

En el segon apartat s'exposa àmpliament l'arquitectura bàsica del sistema, posant èmfasi en els punts en els que aquest projecte es centre. Per altra banda, també detallen les característiques funcionals que la implementació d'aquesta arquitectura permet aconseguir.

Finalment, en el darrer apartat s'entra una mica més en detall en aquells components de l'arquitectura general en el que s'ha centrat la recerca i l'experimentació d'aquest projecte.

3.1. Abast del projecte

Aquest projecte de recerca es basa principalment en una arquitectura per entorns grid proposada en un projecte [4] anterior i orientada a solucionar els problemes que impedeixen l'execució de sistemes DDDAS a gran escala. En aquest projecte, basant-se en la mateixa arquitectura es desenvolupa una solució per a un dels components centrals d'aquest sistema, orientat al tractament de la computació urgent, és a dir, per a facilitar i millorar l'execució de tasques amb uns requeriments de temps exigents.

Els entorns grids, formats de l'aglomeració de múltiples recursos heterogenis distribuïts geogràficament, suposen un ambient poc determinista, tot i que la mateixa comunitat investigadora en ressalta la seva maduresa. Tot i així, cal ser conscient que, avui en dia, encara plantegen alguns reptes i limitacions pel que fa a l'execució de tasques.

En primer lloc, el temps de resposta sol ser una de les magnituds principals que mesuren el rendiment i l'eficàcia d'un entorn grid, a l'hora d'executar aquestes tasques. En un entorn grid, els recursos estan distribuïts en diferents dominis administratius, i cadascun d'ells disposa del seu propi equip de personal que l'administra. A la vegada, cadascun d'aquests

dominis disposa de les seves pròpies directives de seguretat i administració, així com de la seva pròpia política de planificació, que arbitra i gestiona l'accés als seus recursos. El processos que implementen tota aquesta tasca de gestió, administració i planificació afecten a a les tasques en quant demoren l'inici de la seva execució, on aquest retard pot arribar a ser considerable en relació amb el temps d'execució.

En segon lloc, les tasques que envien els diferents usuaris tenen uns requeriments d'execució molt diversos. Aquest requeriments poden fer referència a multitud de factors, entre d'altres:

- l'arquitectura de la màquina
- el sistema operatiu
- la memòria disponible
- l'espai lliure en disc
- una dependència de programari
- un cert nivell de privilegis
- connexió a la xarxa o Internet
- la topologia de la xarxa
- accés a una font de dades concreta

En general, els usuaris no solen ser conscients en la seva totalitat dels requeriments que exigeixen les seves tasques, i solen assumir que si s'executen correctament en un ordinador, també ho faran en els altres, sempre que siguin de característiques similars. El fet és que, quan una d'aquestes tasques s'intenta executar en un entorn que no satisfà els seus requeriments, l'execució és avortada i la tasca retorna al gestor de cues, provocant que hagi de tornar a esperar pels seus recursos. En altres casos, però, si el gestor de cues no es capaç de detectar l'errada, la tasca pot quedar aturada i/o bloquejada al node d'execució, causant un malbaratament innecessari dels recursos.

En tercer lloc, les màquines i dispositius que el conformen no són totalment fiables, poden fallar en qualsevol moment, i seria bo que la infraestructura construïda damunt aquests recursos fos capaç d'abstraure i mitigar l'efecte d'aquestes fallades. Així doncs, aquesta infraestructura ha de proporcionar una certa protecció enfront a les fallades, permetent que les tasques afectades puguin reprendre la seva execució en el menor temps possible, i evitant a tota costa que hagin de tornar a començar l'execució des del principi.

L'ús de màquines virtuals i de les tecnologies de virtualització en les que es basen proporcionen tot un conjunt de funcionalitats i avantatges que ens permeten de construir una arquitectura que permet donar solució a aquestes limitacions:

- **Isolació del maquinari:** Les tecnologies de virtualització permeten abstraure a les màquines virtuals del maquinari físic que les sustenta i, per tant, l'execució de les tasques dins una màquina virtual no es veurà afectada per les característiques del sistema hoste. En aquest escenari, totes les màquines físiques amb un hipervisor present podran servir com a plataforma d'execució per a aquelles tasques que s'executin en una VMI.
- **Multiplexació:** Els hipervisors actuals permeten multiplexar els recursos físics de la màquina per executar més d'una instància de màquines virtual de manera concurrent. Mitjançant aquesta tècnica, és possible executar més d'una tasca en el mateix node d'execució, esprement encara més el potencial de superposar el còmput amb les comunicacions.
- **Cloud computing:** Les màquines virtuals poden, de manera anàloga a les màquines físiques, integrar un entorn d'execució que les habiliti per a funcionar com a nodes d'execució, amb les mateixes capacitats i habilitats que les màquines físiques. A més a més, és possible organitzar un conjunt d'aquestes màquines virtuals de manera que sigui possible aplicar-hi una configuració especial o dotar-les d'unes polítiques de prioritat diferents. D'aquesta manera es pot potenciar la reutilització de les instàncies, ja que la finalització del treball pel qual fou instanciada una màquina virtual no té perquè implicar la seva finalització, sinó que pot romandre a l'espera de noves tasques, estalviant el sobrecost de tornar a aixecar una nova instància.

Com s'havia introduït, aquestes tres característiques permeten solucionar i mitigar en gran part les limitacions observades amb anterioritat.

En primer lloc, el temps de resposta de les tasques, que és una magnitud d'importància cabdal en els treballs de computació urgent, es pot millorar gràcies a la construcció de subconjunts de màquines virtuals amb unes polítiques d'execució especialitzades, fins al punt de poder-se dedicar en exclusiva a l'execució d'aquests treballs prioritaris.

En segon lloc, la problemàtica dels requeriments insatisfets de les aplicacions desapareix. Una vegada es disposa d'una VMI amb els requeriments necessaris per a l'execució d'un treball, nomes caldrà assegurar-se que aquest treball s'executi sempre sobre una instància d'aquesta VMI.

En tercer lloc, si les capacitats de la màquina física i de l'hipervisor permeten l'execució de més d'una màquina virtual, aleshores es podrà assolir la multiplexació d'un node físic. D'aquesta manera és possible millorar el rendiment i la productivitat dels recursos.

Tot i així, la introducció de les tecnologies de virtualització i l'ús de les màquines virtuals, a més de donar solució a la majoria de limitacions, també afegeixen certs reptes i complexitats que caldrà estudiar i prevenir els possibles efectes col·laterals, com la pèrdua de rendiment derivada de la sobrecàrrega de la virtualització. Aquests i altres aspectes seran tractats més endavant els capítols següents.

3.1.1. Anàlisi de requeriments i casos d'ús

La implementació d'aquesta idea imposa ben pocs requeriments a sobre de la plataforma física dels recursos computacionals, sempre que la instanciació d'una màquines física en un node d'execució es pugui tractar i gestionar com si fos qualsevol altre tasca provinent d'un usuari. A més a més, si la VMI és capaç d'executar un gestor de cues locals i, durant l'arrancada, unir-se a una instal·lació existent com a node d'execució, ja estarà llesta per a rebre i executar treballs.

Una vegada analitzat l'abast d'aquest projecte, ja es poden enumerar els principals requeriments que són imprescindibles per a dur-lo a terme:

1. **Suport per virtualització als nodes d'execució:** Els nodes d'execució s'han de tenir instal·lat un hipervisor i han de ser capaços d'executar màquines virtuals. Existeixen moltes implementacions alternatives d'hipervisors que, en conjunt, permeten donar suport a la quasi totalitat del mercat dels computadors actuals.
2. **Gestor de cues local amb suport per a treballs VMI:** El gestor de cues local que s'utilitzi ha de ser capaç d'executar màquines virtuals com si fossin treballs estàndards. Això implica que el gestor de cues ha de comunicar-se amb l'hipervisor per a poder executar les VMI. Actualment ja existeixen alguns gestors de cues locals amb suport per a màquines virtuals, com en el cas de Condor.

Satisfets aquests requeriments, podem enumerar els casos d'ús a que es pretén donar servei amb aquesta arquitectura, el disseny de la mateixa es descriu en l'apartat que ve a continuació:

- **Treballs legacy:** Anomenarem així aquells tipus de treballs estàndards que no requereixen cap tipus de funcionalitat derivada de la introducció de les tecnologies de virtualització, però que per compatibilitat evident han de poder seguir-se executant de manera tradicional sobre els recursos físics del grid. Aquests treballs no requeriran de cap màquina virtual com a suport a la seva execució i seran gestionats directament pel gestor de cues local del lloc.
- **Instanciació de VMI:** En aquesta arquitectura, les màquines virtuals es poden utilitzar per oferir certes funcionalitats i avantatges a aquells treballs que les utilitzen com a plataforma d'execució. Per tal que els treballs dels usuaris s'hi puguin executar però, abans caldrà preparar-les o implantar-les de manera adient, instal·lant-hi un entorn d'execució adequat. Finalment, les màquines virtuals implantades seran tractades com si fossin treballs, i transmeses a una màquina física per a la seva instanciació.
- **Instanciació de treballs:** Els treballs dels usuaris tenen associada la màquina virtual en la que s'han d'executar. Si existeixen instàncies disponibles no ocupades d'aquestes màquines, els treballs hi seran lliurats per a ser executats. Si, per altra banda, no hi ha cap màquina virtual compatible lliure, el sistema s'encarregarà d'instanciar les còpies necessàries i, posteriorment, redirigir-hi els treballs. El sistema gestionarà i farà el seguiment durant tota l'execució dels treballs, fins a assegurar que l'usuari n'ha rebut els resultats correctament.

3.2. Arquitectura bàsica

Existeixen projectes anteriors que ja han fet recerca sobre les arquitectures i sobre els serveis que han d'oferir. Un dels projectes més influents en aquesta àrea ha estat In-VIGO [17], de la Universitat de Florida. El seu objectiu és proporcionar un entorn distribuït basat en màquines virtuals on els usuaris finals poguessin executar múltiples aplicacions, amagant tota la complexitat subjacent. El seu enfocament és afegir tres capes de virtualització al model tradicional de computació en grid.

A la figura 3 es representa un esquema de l'arquitectura típica d'un grid. En ella apareixen els diversos elements que la formen i on cadascun d'ells té assignat un conjunt de responsabilitats ben concretes:

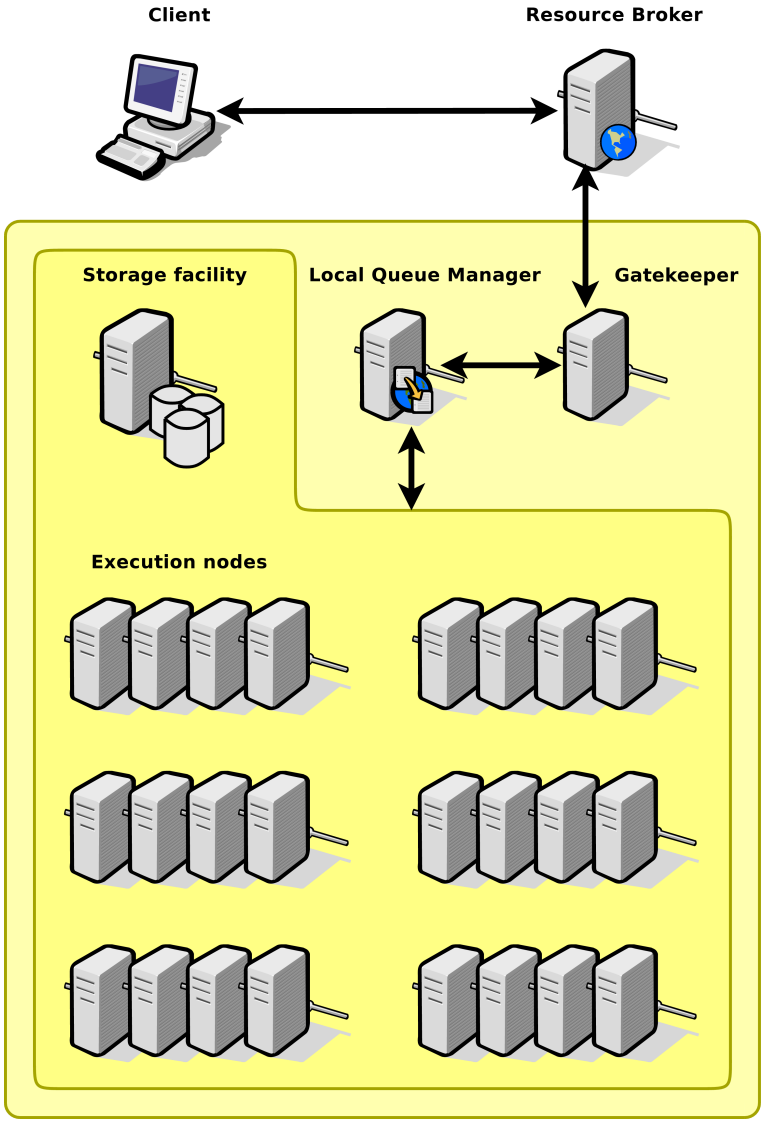

Figura 3: Arquitectura típica d'un grid

- **Resource Broker (RB):** Aquest element és compartit per a tots els llocs que formen un grid. La responsabilitat d'aquest element és la d'identificar i classificar els recursos disponibles als diferents grids, així com de seleccionar i assignar els més adients per executar cada tasca. Els recursos dels diferents grids s'administren localment i el seu accés està arbitrat mitjançant diferents polítiques. El *Resource Broker* pren les seves decisions de manera independent, sense estar subjecte a cap mecanisme de control global, i únicament basades en la informació que cada grid.
- **Gatekeeper (GK):** S'encarrega de comunicar-se i negociar amb l'RB el nombre i característiques de recursos disponibles en un lloc determinat, a més de publicar

informació sobre la seva disponibilitat. El GK exerceix d'intermediari entre l'RB i el gestor de cues local a l'hora de redirigir les tasques que s'envien per ser executades.

- **Gestor local de cues (LQM):** Gestiona els recursos locals i hi planifica l'execució de les tasques segons la seva prioritat.
- **Emmagatzemament:** Conjunt d'elements que solen estar destinats en exclusiva a l'emmagatzemament i persistència de conjunts de dades i informació que són utilitzats per la resta d'elements del lloc. Entre aquesta informació podem trobar les entrades i sortides dels treballs, les imatges virtuals de les màquines, o la seva configuració, entre d'altres. Totes aquestes dades es solen fer accessibles mitjançant un sistema de fitxers compartit.
- **Nodes d'execució:** Conjunt de màquines físiques que serveixen com a plataforma d'execució dels treballs enviats.

A partir d'aquesta arquitectura, i tenint en compte els requeriments descrits amb anterioritat, cal adaptar-la per a poder satisfer els objectius que s'han plantejat. Principalment,.caldrà afegir els components necessaris per a poder gestionar màquines virtuals com a nodes d'execució. A la figura 4 es mostra aquesta arquitectura ampliada, amb les modificacions següents:

- Part dels nodes d'execució ara estan formats per màquines virtuals, que han estat instanciades com a treballs a sobre d'una part dels nodes físics. Aquests nous nodes virtuals s'organitzen com un nou conjunt de recursos accessibles de manera completament transparent mitjançant el gestor local de cues.
- El GK integra un nou servei encarregat de gestionar aquest nous nodes d'execució virtuals. Serà responsabilitat seva discriminar els treballs i redirigir-los a un node virtual existent, instanciar-ne un sota demanda, o bé dirigir-los al gestor de cues per a ser executats en una node físic.

A partir d'aquesta arquitectura adaptada a la presència de màquines virtuals, a l'apartat següent es descriuen les característiques funcionals de què disposa aquesta arquitectura.

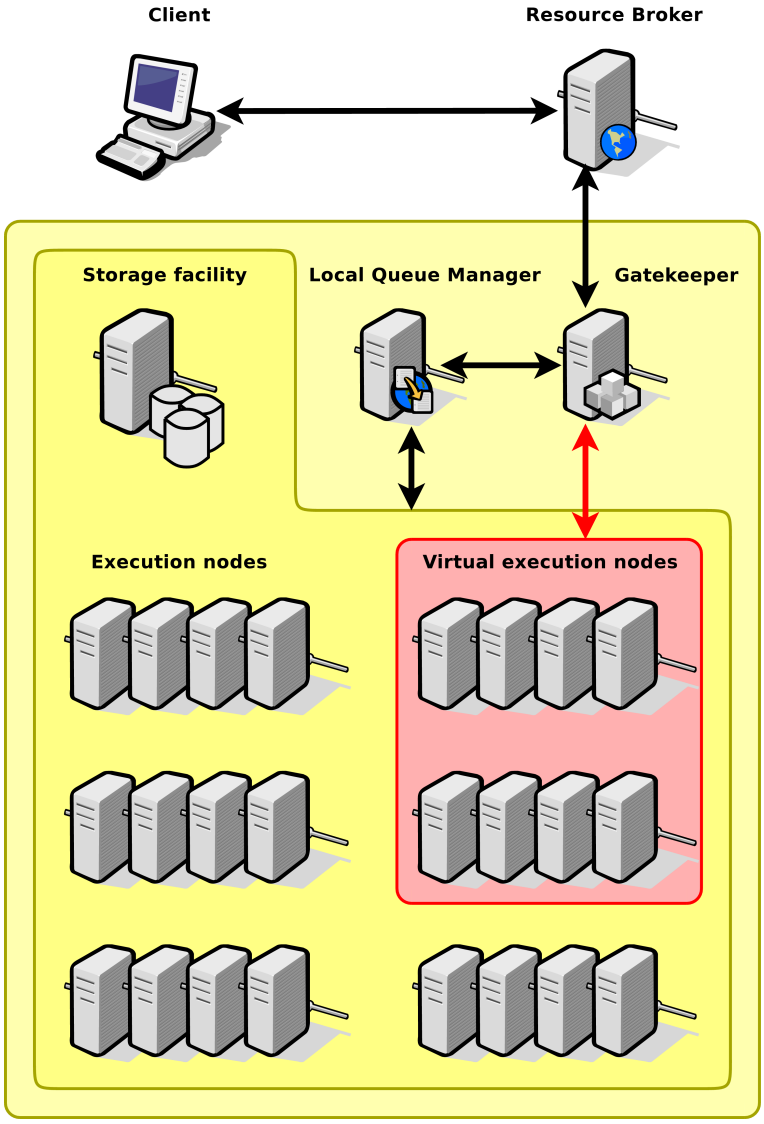

Figura 4: Ampliació de l'arquitectura amb les màquines virtuals

3.2.1. Característiques funcionals

La figura 4 mostra l'esquema de l'arquitectura adaptada, l'objectiu de la qual és assolir les especificacions següents:

- Els treballs estàndards o *legacy*, han de poder executar-se sense cap tipus de modificació i de manera completament transparent a l'usuari. Aquests treballs són els que no requereixen d'una màquina virtual com a plataforma d'execució i, en el seu lloc, s'executaran directament sobre els nodes d'execució físics tradicionals.
- Els treballs que requereixen una màquina virtual com a plataforma d'execució seran gestionats també de manera transparent. L'usuari no percebrà la complexitat ni les

accions que es duran a terme per sota per tal de desplegar les instàncies de màquines virtuals necessàries per a executar aquests treballs. Aquests treballs es comportaran de manera anàloga amb els treballs estàndards, i l'usuari en rebrà els resultats de la mateixa manera.

- El sistema disposarà d'un conjunt de VMI que podran servir de plataforma d'execució als diferents treballs. Aquest conjunt podrà ser ampliat per les màquines virtuals que proporcioni l'usuari.
- La infraestructura ha d'adaptar-se dinàmicament a la carrega de sistema, afegint noves instàncies de màquines virtuals si el nombre de treballs o la longitud de la cua així ho requereix, segons uns criteris prèviament establerts.
- Els treballs urgents han de rebre un tractament adequat que permeti obtenir els resultats en un temps ajustat. L'ús de polítiques de prioritat ha d'anar destinada a reduir els seus temps de resposta.
- Les instàncies de les màquines virtuals han de ser reutilitzades en aquells casos en els que sigui factible. Una bona política de reciclatge de les instàncies és clau per a estalviar temps i processament. Per aquest motiu, serà imprescindible que el sistema disposi de criteris per a avaluar i detectar el moment idoni en el una màquina pot ser aturada.

Aquestes característiques desitjades cal implementar-les sense afectar de manera radical a l'arquitectura subjacent i, per tant, serà necessari que els canvis no siguin invasius. A més, cal evitar tota interferència amb l'execució tradicional dels treballs del grid. A la figura 5 es mostra un esquema d'aquest nou servei integrat al Gatekeeper que s'encarrega de la gestió de les màquines virtuals.

En aquest esquema trobam un únic punt d'entrada destinat als administradors i orientat a les tasques de control, supervisió i manteniment del servei. Des d'aquest punt també es controla el subsistema VMI, permetent la consulta, accés i emmagatzemament de noves imatges de màquines virtuals, per part dels usuaris autoritzats.

També és present un servei proveïdor d'informació, que agrega dades provinents tant del punt d'entrada d'administrador com del servidor de VMI, i on la seva responsabilitat és oferir informació sobre l'estat del sistema, dels treballs en execució, dels recursos disponibles i del

nivell d'ocupació dels mateixos. Per altra banda, també ofereix informació detallada sobre les capacitats i característiques de les VMI emmagatzemades i que poden ser usades com a plataforma d'execució dels treballs.

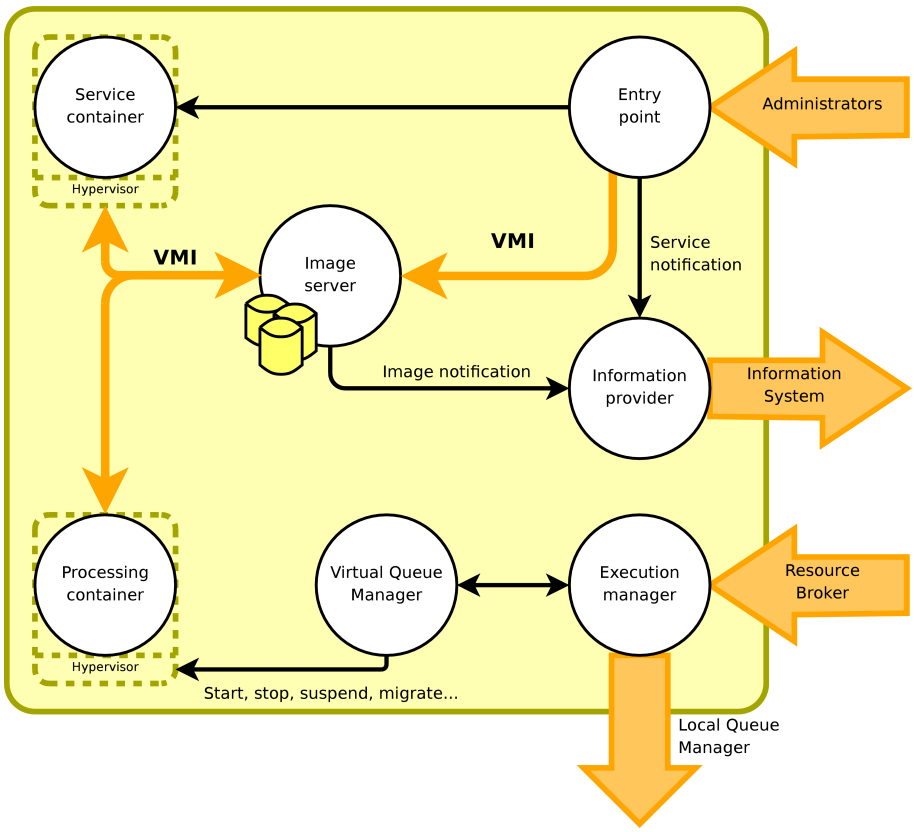

Figura 5: Esquema del servei de suport a la virtualització

Els components principals d'aquesta arquitectura, almenys pel que fa al nombre d'ells, són els nodes d'execució. Els que es mostren a l'esquema són només aquells nodes d'execució virtuals, és a dir, aquells que poden executar màquines virtuals mitjançant el seu hipervisor instal·lat. Aquests nodes d'execució virtuals s'han subdividit, al seu lloc, en dues categories: Per una banda els contenidors de còmput són els que s'utilitzen de manera anàloga als nodes físic, és a dir, són els que s'utilitzen per als treballs tradicionals. Per altra banda els contenidors de servei són aquelles màquines virtuals que allotgen treballs amb característiques especials, bé per la seva durada, prioritat o urgència, entre d'altres.

Els usuaris del grid no aprecien cap diferència a l'hora d'accedir a aquests recursos virtuals, ja que apareixen de manera completament transparent a la interfície que fan servir. El *Resource Broker* és l'encarregat de proporcionar a aquesta interfície tota la informació sobre

l'estat i disponibilitat dels recursos, així com la càrrega de treballs i la ocupació de les cues. Les tasques que enviïn els usuaris passen pel gestor d'execució, que és l'encarregat de determinar com s'hauran d'executar, o bé mitjançant un recurs físic a través del gestor de cues local, o bé en un recurs virtual mitjançant el gestor de cues virtuals. Depenent de la configuració del grid, un sol gestor de cues pot gestionar els dos tipus de treballs, tant els físics com els virtuals.

Finalment, el gestor de cues virtual disposarà de la independència i de la informació necessària com per a prendre decisions sobre les instàncies de màquines virtuals, des de la seva arrancada, implantació, suspensió, aturada o migració.

3.3. Arquitectura específica

La recerca en aquest projecte s'ha centrat en aquells components que s'encarreguen de la gestió de les màquines virtuals i dels treballs que s'hi executen. En general, l'àmbit d'aquest projecte intenta abastar la totalitat del cicle de vida d'aquestes màquines i treballs: des de la seva implantació i instanciació, l'execució i seguiment de les tasques, fins a la obtenció dels resultats i posterior aturada o reciclatge de les instàncies. Els components que s'han desenvolupat s'emmarquen dins un subsistema anomenat gestor d'espais d'execució virtuals, que integra diversos elements de l'arquitectura definida a l'apartat anterior. A la figura 6 es pot observar un diagrama amb l'estructura més detallada d'aquests components.

En aquesta estructura, les peticions de l'exterior per executar treballs entren a través d'un punt d'entrada al middleware del grid, i allà es discrimina si són treballs *legacy* o si són treballs que requereixen de la seva execució en un espai virtual. Els treballs *legacy* o tradicionals es redirigeixen al gestor de cues local i, a partir d'aquest moment, el seu tractament serà idèntic al que rebrien en un entorn grid no virtualitzat. Per altra banda, aquells treballs que requereixin d'un espai virtual per a la seva execució seran redirigits al gestor d'execució. Aquest gestor disposa d'una cua per anar emmagatzemant els treballs que li van arribant, mentre negocia l'assignació dels seus recursos i planifica la seva execució.

Els treballs que requereixen d'un espai virtual d'execució s'executen sobre instàncies de màquines virtuals generades a partir d'unes imatges (VMI) que, o bé ja estaven disponibles en un repositori del sistema, o bé les proporciona el mateix usuari annexades al treball, segons les seves necessitats. En qualsevol cas, aquestes imatges s'han de preparar per tal que puguin

servir com a plataforma d'execució. A aquest procés de preparació se l'anomena implantació i és responsabilitat del gestor de màquines virtuals. Per a aquest projecte s'han estudiat i implementat dos mecanismes d'implantació que per tal d'assegurar que aquestes imatges compleixen tots els requeriments necessaris per a poder ser instanciades i que s'integrin en els recursos del grid. Aquests dos mecanismes d'implantació, que s'expliquen al capítol següent, donen servei a la majoria de casos d'ús existents i abasten quasi tots els tipus d'imatges virtuals existents.

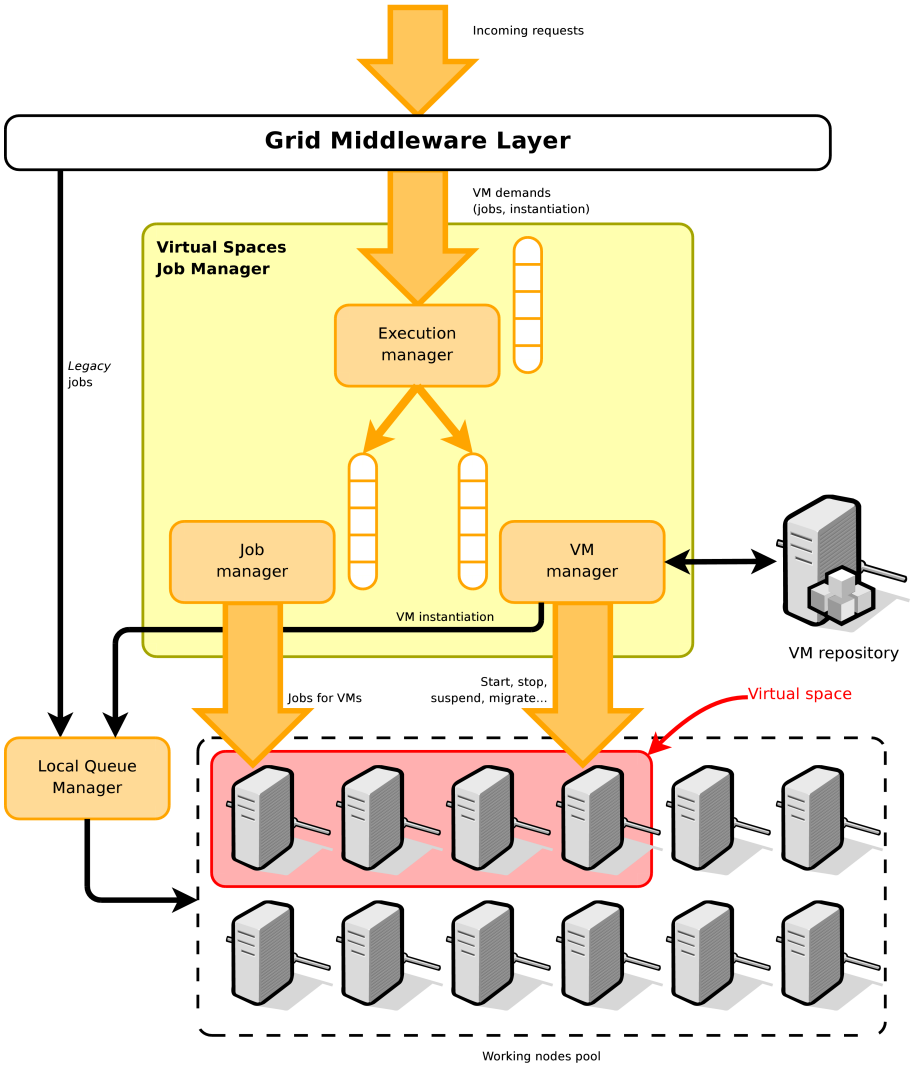

Figura 6: Arquitectura específica dels components desenvolupats

Aquest mateix gestor també es responsabilitza de la resta de processos del cicle de vida de les màquines virtuals, com l'aturada, la suspensió temporal o la migració cap a un altre node físic. En el cas de l'aturada de les màquines virtuals, és important determinar el moment

adient per dur-la a terme i prioritzar el reciclatge de les instàncies per a altres treballs. Per això, el gestor de màquines virtuals disposa d'un conjunt de criteris que permeten decidir el moment idoni per a que aquesta aturada afecti el menys possible a la resta de tasques. De manera addicional, també s'han estudiat els efectes de la suspensió i la migració de màquines virtuals, com a operacions que poden substituir a l'aturada.

Finalment, comentar que totes les imatges de màquines virtuals disposen d'un codi que les identifica de manera única i que permet fer-ne referència a través de tot el sistema. D'aquesta manera, la relació entre el treball i la màquina virtual en la que ha de ser executat es pot establir utilitzant unes directives especials en els seus fitxers de descripció. La figura 7 es pot observar un diagrama d'aquest mecanisme.

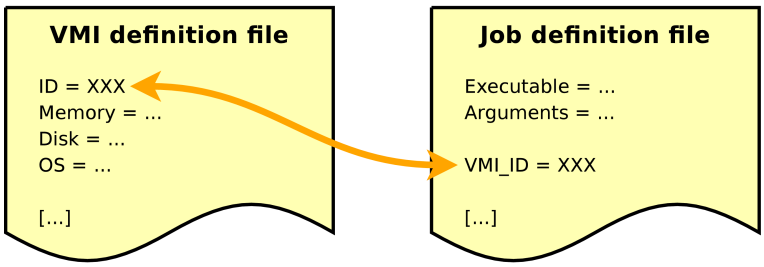

Figura 7: Fitxers de definició de les imatges i els treballs

El proper capítol explica amb més detall la recerca i el desenvolupament que s'han dut a terme en alguns components concrets, així com la seva interrelació amb la resta de subsistemes d'aquesta arquitectura.

4. Implementació i experimentació

Al capítol anterior s'ha descrit una arquitectura per donar suport a l'execució de tasques sobre màquines virtuals, utilitzant el concepte d'espais d'execució virtuals, i posant èmfasi en aquells components concrets de l'arquitectura en els que s'ha centrat la recerca. Concretament, la implementació s'ha centrat en el subsistema del gestor d'espais d'execució virtuals. La responsabilitat principal d'aquest component és gestionar els treballs i les màquines virtuals durant tot el seu cicle de vida.

El cicle de vida d'una màquina virtual inclou tres etapes principals (arrancada, execució i aturada) i tres processos opcionals (implantació, suspensió i migració). En aquest capítol, en el primer apartat s'exposen els diferents tipus d'imatges que poden donar lloca a una màquina virtual. En el segon apartat s'entra amb detall en el procés d'implantació d'una VMI, que suposa la seva preparació per a l'arrancada, descrivint els diferents mecanismes amb que es pot dur a terme. En el tercer apartat s'analitzen els criteris i arguments que condueixen la màquina virtual a la tercera etapa, la d'aturada. I, finalment, al quart apartat es fa un petit estudi sobre els avantatges i inconvenients dels processos de suspensió i migració, així com de quina manera afecten al rendiment d'aquestes màquines en un entorn grid virtualitzat.

4.1. Tipologia d'imatges de màquines virtuals

En aquest sistema, els usuaris que envien les seves tasques les acompanyen de la imatge de màquina virtual (en endavant, VMI) sobre la qual volen que siguin executades. Aquestes VMI que serveixen com a plataforma d'execució han d'estar correctament configurades o, en cas contrari, l'execució de les tasques no serà possible. El requisit indispensable per a possibilitar aquesta execució automatitzada de les tasques és que la VMI disposi d'un gestor de cues correctament configurat, de manera que en arrancar la imatge aquesta s'uneixi a un gestor de cues central i aparegui com a un nou node d'execució. Serà en aquest moment quan les tasques que l'usuari ha enviat, conjuntament amb la VMI, es podran anar executant.

L'escenari en què les VMI proporcionades pels usuaris ja estan preparades per a executar les seves tasques és un tant utòpic. No es pot suposar que els usuaris disposin dels coneixements necessaris per a satisfer aquests requeriments i, per tant, és probable que moltes de les VMI que proporcionin no els satisfacin. Així doncs, en aquests casos el sistema ha

d'intervenir per tal de dur a terme una operació d'implantació d'un entorn d'execució adequat per tal de possibilitar l'execució de les tasques.

Les VMI que proporcionen els usuaris poden estar confeccionades de maneres molt diverses, i cal tenir en compte tots aquells aspectes que poden afectar al procés d'implantació. A continuació es detallen aquestes característiques més rellevants i que afecten més a l'hora de dur a terme aquesta operació.

- **Presència d'un gestor de cues no adequat:** En alguns casos la màquina virtual ja disposa d'un gestor de cues, tot i que aquest pot estar mal configurat, pot no complir els requeriments necessaris, o pot no ser compatible amb el gestor de cues central. En aquests casos, durant la implantació caldrà adaptar o substituir la instal·lació d'aquest gestor de cues per una instal·lació que sí compleixi els requeriments.
- **Nucli encastat o proporcionat externament:** L'arrancada de la màquina virtual es pot dur a terme de dues maneres: o bé utilitzant algun dels nuclis que ja tengui instal·lats al seu interior, o bé proporcionant-ne un d'extern, el qual es podrà complementar amb una imatge de RAM inicial i paràmetres addicionals. Segons el sistema d'arrancada, es poden aplicar diferents tècniques d'implantació, tal com es veurà al punt següent.
- **Configuració de l'emmagatzemament:** L'estructura dels sistemes de fitxers adjunts a una VMI és fonamental per a dur a terme el procés d'implantació amb èxit. En la majoria dels casos, els sistemes de fitxers es solen desar en arxius imatge que representen, o bé una partició individual, o bé un disc sencer. En altres casos, però, els sistemes de fitxers poden estar gestionats per un nivell d'abstracció superior com un gestor de volums lògics o, fins i tot, poden estar xifrats. Una mateixa VMI pot combinar sistemes de fitxers expressats en qualsevol d'aquestes maneres i, depenent del mecanisme d'implantació, totes s'han de tenir en compte.
- **Dependències de programari:** La plataforma operativa i el programari instal·lat en cada VMI és molt heterogeni i no es pot assegurar l'existència de cap programari o versió determinada. És per això que el procés d'implantació també haurà de tenir en compte que, conjuntament al gestor de cues, també haurà d'assegurar que la màquina virtual disposi de les dependències de programari necessàries per a executar-lo.

4.2. Mètodes d'implantació

Com s'ha vist al punt anterior, hi ha molts aspectes en la confecció d'una VMI que poden influir en un procés d'implantació. Aquesta operació d'implantació té com a objectiu assegurar que les màquines virtual disposen d'un entorn adequat que hi possibiliti l'execució de les tasques de l'usuari. Però l'heterogeneïtat en les característiques de les VMI fa que no existeixi un mètode d'implantació universal i obliga a plantejar diferents mètodes alternatius per poder tractar amb èxit el conjunt més ampli possible de VMI.

En aquest apartat es descriuen alguna de les tècniques existents per a dur a terme el procés d'implantació i configuració d'un gestor de cues com a entorn d'execució sobre una VMI. La primera de les tècniques fa ús dels paràmetres d'arrancada del nucli Linux per injectar una secció de codi que farà la implantació tot just la màquina virtual arranqui. La segona tècnica duu a terme la implantació de l'entorn d'execució accedint directament al sistema de fitxers de la VMI.

4.2.1. Mètode init

L'arrancada d'un sistema Linux consta de múltiples etapes [E19], tal i com es mostren a la figura 8. En la primera d'elles, la BIOS inicialitza el maquinari bàsic i cerca el dispositiu des del qual arrancar el sistema operatiu. Una vegada trobat i inicialitzat aquest dispositiu, passa el control al carregar d'arrancada. Existeixen multitud de carregadors d'arrancada diferents. En el món de les distribucions GNU/Linux, els dos més emprats són GRUB [E18] i LILO, i pel cas d'aquest projecte ens centrarem d'ara endavant en l'arquitectura de GRUB, ja que és un dels carregadors d'arrancada més potents, flexibles i portables, i és el que s'instal·la per defecte en la gran majoria de distribucions GNU/Linux.

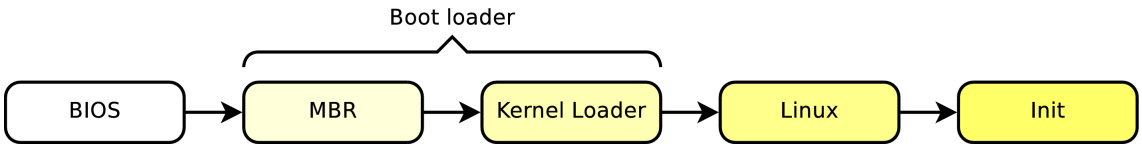

Figura 8: Procés d'arrancada d'un sistema GNU/Linux

El carregador d'arrancada està format per dues fases, la primera és l'MBR i és la responsable d'analitzar la taula de particions, seleccionar la partició activa i passar el control a la segona fase. La segona fase de GRUB és capaç de reconèixer els sistemes de fitxers,

carregar la llista de nuclis disponibles juntament amb la configuració corresponent de cada un d'ells i, finalment, executar-los. En la darrera etapa de totes, una vegada el nucli Linux ha acabat d'inicialitzar tota la resta de maquinar i muntar els sistemes de fitxers, la seva darrera responsabilitat és iniciar el primer procés del sistema operatiu, el procés *init*, que s'encarrega d'engegar tota la resta de processos o aplicacions de l'espai d'usuari que conformaran tot l'entorn de serveis del sistema operatiu.

El nucli Linux accepta un ampli conjunt de paràmetres que, especificats a l'hora d'arrancar-lo, permeten definir el seu comportament i modificar alguns aspectes del seu funcionament. Un d'aquests paràmetres permet especificar l'aplicació que cal llançar al final del procés d'arrancada (el procés *init*). Aquest paràmetre és la ruta a un fitxer executable ja present a la VMI, que en la majoria dels casos serà una implementació alternativa del procés *init* estàndard, com es pot observar a la figura 9. Però la qüestió més interessant d'aquesta tècnica és que se li poden especificar arguments addicionals a aquest executable, de manera que si com a executable s'especifica un intèrpret de comandaments, mitjançant aquests arguments addicionals es poden especificar el conjunt de comandes que volem que s'executin durant l'arrancada de la VMI, i que duran a la terme la implantació del gestor de cues desitjat.

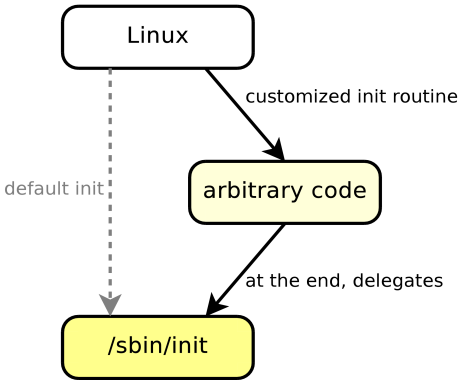

Figura 9: Implementació init alternativa

Funcionament

La base d'aquest primer mecanisme d'implantació rau en la possibilitat d'injectar una secció arbitrària de codi que sigui capaç de detectar la presència o no d'un entorn d'execució adequat i d'assegurar-se'n que estigui correctament configurat o, en cas contrari, de dur a terme les operacions necessàries per a que així sigui. En la majoria de sistemes Linux, existeix tota una estructura de fitxers i directoris que especifiquen el conjunt i seqüència de programes

i serveis que s'executen en iniciar-se el sistema. Aquesta estructura es troba a */etc/rcN.d*, on N indica el nivell d'execució de la sessió actual, i es correspon a l'especificació del sistema d'arrancada del System V, una versió de UNIX desenvolupada per AT&T i llançada el 1983.

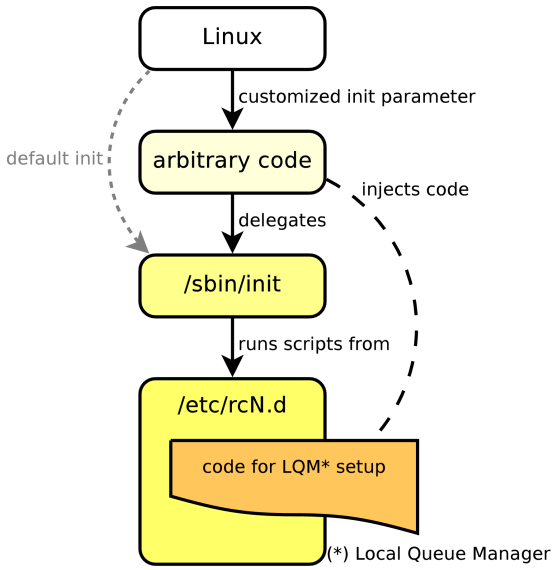

Figura 10: Injecció del codi necessari per a la implantació

Per tant, si s'aconsegueix injectar codi a la ubicació apropiada d'aquesta estructura, es pot aconseguir que aquest codi s'executi com a part de la seqüència de rutines d'inicialització de l'entorn d'usuari, de manera que es pugui assegurar que el gestor de cues ja funciona abans de passar el control a l'usuari.

Aprofitant els paràmetres d'arrancada del nucli Linux, es pot especificar un intèrpret de comandaments com a comanda *init* i, mitjançant paràmetres addicionals, a aquest intèrpret se li pot especificar el conjunt de comandes que volem que sigui executat. Tot i que la idea darrera d'aquesta tècnica és ben senzilla, a la pràctica ha presentat algunes limitacions que ha calgut esmenar per poder demostrar la viabilitat d'aquest mecanisme.

- **Caràcters especials:** La primera limitació que se'ns presenta en aquest mètode és com proporcionar la secció de codi a través dels paràmetres, ja que la presència d'alguns caràcters especials al codi embulla a la funció que avalua els paràmetres i impedeix que els reconegui correctament. Es fa necessari, doncs, escapar de manera adequada tots els caràcters susceptibles de causar algun tipus de problema, com les cometes o els salts de línia, entre molts altres. Aquesta operació d'escapament introdueix un pas

addicional de transcripció per tal d'aconseguir una versió del codi en la forma adequada per a ser utilitzada com a paràmetre i, a més a més, n'augmenta considerablement la mida a la par que en dificulta la llegibilitat.

- **Longitud limitada:** La segona limitació és que la longitud total dels paràmetres que es poden especificar al nucli a l'hora de l'arrancada és limitada, i en les versions més recents el seu valor màxim es troba en 2048 caràcters [E25]. Aquesta longitud és insuficient per especificar tot el codi i, a més, cal deixar cert marge per a altres paràmetres que també poden ser necessaris per arrancar de manera correcta el sistema. És per això que cal dividir el codi destinat a la implantació en dues parts. La primera part s'assegurarà de situar una secció de codi a l'estructura */etc/rcN.d* per tal que arranqui en engegar el sistema. Aquesta primera secció de codi serà la responsable de descarregar la segona part des de la xarxa i d'executar-la. D'aquesta manera, la segona secció de codi pot tenir una mida arbitràriament gran, ja que no ve especificada mitjançant la línia de comandes del nucli.

Una vegada esmenades aquestes limitacions, la resta de requeriments que són necessaris per a dur a terme amb èxit la implantació són molt més senzills d'assolir.

- **Accés d'escriptura a la partició arrel:** Per tal de poder injectar el codi necessari per a dur a terme la implantació del gestor de cues, és necessari que el nucli Linux munti el sistema de fitxers arrel amb permisos d'escriptura durant les primeres etapes de l'arrancada del sistema. Aquest comportament es pot aconseguir especificant un paràmetre addicional «*rw*» al nucli.
- **Presència de l'intèrpret adient:** En aquest primer mecanisme d'implantació, el codi responsable de la implantació del gestor de cues a les VMI ha de ser interpretat dins l'entorn de la màquina virtual. Això fa que el codi sigui dependent de l'intèrpret pel qual es dissenya i, encara més important, requereix que aquest intèrpret estigui disponible dins la màquina virtual. Per aquest projecte de recerca, aquest codi s'ha desenvolupat pensant en ser executat per qualsevol intèrpret de comandaments compatible amb l'estàndard POSIX [E20]. Un exemple d'intèrpret de comandaments compatible amb aquesta especificació és BASH [E21], que actualment ve muntat per defecte en la gran majoria de distribucions GNU/Linux.

Per altra banda, si les funcionalitats del llenguatge de l'intèrpret de comandes POSIX no fossin suficients per a les necessitats de la implantació, es podria estudiar emprar altres intèrprets més potents com Perl [E22] o Python [E23]. Aquests intèrprets ofereixen un joc molt més complet de funcionalitats, però la probabilitat de trobar-se instal·lats a la VMI no és tan elevada com per a un intèrpret POSIX.

- **Suport per arrancada System V:** El codi que duu a terme la implantació de l'entorn d'execució s'injecta com a part del sistema d'arrancada del System V [E24], que es correspon a l'estructura de directoris */etc/rcN.d*. La gran majoria de distribucions GNU/Linux i derivats UNIX utilitzen l'especificació System V o una de compatible, tot i que hi ha algunes excepcions, com els derivats BSD. En aquests darrers casos es podria modificar la primera secció de codi per a que s'injecti en una ubicació anàloga del seu sistema d'arrancada, com */etc/rc.local* o */etc/rc.d*.
- **Accés a la xarxa:** Com ha s'ha comentat en la descripció d'aquesta tècnica, part del codi responsable de dur a terme la implantació cal descarregar-lo des de la xarxa. Per tant, és imprescindible que la VMI tengui accés a la xarxa per poder accedir tant a la segona secció de codi, com als fitxers necessaris per a instal·lar el gestor de cues.

Avantatges i inconvenients

Aquest primer mètode d'implantació presenta unes característiques especials que el fan molt adequat per a certs tipus d'escenaris, tot i que en alguns la seva utilització no és factible. A continuació es detallen els principals avantatges i inconvenients que s'han detectat amb l'anàlisi d'aquest mètode.

- ✔ **Implantació remota i distribuïda:** La manera particular de funcionar d'aquest mecanisme permet implantar el gestor de cues de manera remota en qualsevol VMI , sempre que disposi de connexió a la xarxa. Aquesta característica és molt atractiva en aquells casos en els que el cost de transmetre la VMI per a dur a terme la implantació sigui inviable (bé per la mida excessiva de la imatge o bé perquè l'amplada de banda disponible sigui insuficient. En aquests casos, caldrà tenir una còpia dels fitxers d'instal·lació del gestor de cues accessible a través de la xarxa local.
- ✘ **Consum lineal d'amplada de banda:** Tal com ja s'insinua al paràgraf anterior, l'ús d'aquesta tècnica només elimina la transferència de les VMI en els casos en els que

aquestes es trobin ja al sistema hoste. Per altra banda, el que fa es distribuir els requeriments d'amplada de banda cap als nodes d'execució, ja que cada VMI que s'instanciï i necessiti ser implantada haurà de descarregar-se una còpia del gestor de cues des de la xarxa. Per tant, i com ja s'ha comentat, seria molt convenient que els nodes d'execució disposessin d'una copia a la seva xarxa local d'aquests recursos.

Aquesta tècnica d'implantació presenta, doncs, un consum lineal d'amplada de banda ja que, mentre una implantació centralitzada només cal fer-la una vegada per VMI, aquesta tècnica requereix implantar totes les còpies que s'instanciïn en qualsevol node d'execució.

- **Només per a nucli proporcionats externament:** Ja s'ha descrit que aquesta tècnica es basa en subministrar paràmetres addicionals al nucli per a dur a terme la implantació. Però per a poder especificar aquests paràmetres, el nucli s'ha de proporcionar externament i aquest, o bé ha disposar de tots els mòduls necessaris encastats, o bé aquests mòduls han d'estar disponibles a la VMI. Això suposa una pèrdua de flexibilitat, ja que redueix el conjunt de VMI sobre el que es pot aplicar la tècnica.

Proves de rendiment

Per a comprovar la viabilitat d'aquest mecanisme ha calgut verificar que es complissin dues condicions. Per una banda que fos viable injectar el codi necessari per a fer la implantació mitjançant els arguments del nucli, i per l'altra que aquesta injecció de codi no tengués altres efectes col·laterals que poguessin provocar un funcionament incorrecte de la màquina virtual.

Amb l'objectiu de calcular el rendiment d'aquest primer mecanisme d'implantació s'ha dissenyat el següent experiment, consistent en mesurar la durada de les diferents etapes del procés d'implantació, i la seva participació en el temps total d'arrancada de la màquina virtual. Totes les proves s'han dut a terme sobre la mateixa màquina, un ordinador HP amb 1 GiB de memòria RAM i dos processadors Intel Pentium 4 funcionant a 2'6 GHz. L'hipervisor de màquines virtuals seleccionat ha estat Xen, en la seva versió 3.2.1, i el gestor de cues local seleccionat ha estat Condor. La VMI a implantar ha estat una Debian Lenny 5.0, construïda mitjançant l'eina *debootstrap*, pròpia de Debian, que, al no incorporar cap nucli encastat, caldrà proporcionar-ne un d'extern a l'hora d'arrancar-la. Els seus sistemes de fitxers es

proporcionen en forma d'arxius individuals que simulen ser diferents particions. Per altra banda, la instal·lació base de Debian Lenny ja inclou un intèrpret POSIX de sèrie, tot i que no inclou cap gestor de cues i, amb tota seguretat, tampoc inclou les dependències necessàries per a poder-lo executar.

Per tal d'augmentar la confiabilitat i representativitat dels resultats, s'han dut a terme una dotzena d'execucions, buidant la memòria cau del disc entre execució i execució per simular les mateixes condicions inicials. A l'hora de mesurar el rendiment d'aquest mecanisme d'implantació, s'ha dividit el procés d'arrancada i implantació de la VMI en vuit etapes. Les tres primeres es corresponen al propi procés d'arrancada d'un sistema GNU/Linux estàndard mentre que les cinc etapes restants es corresponen al procés d'implantació en sí. A continuació es descriu amb més detall les característiques i funcions principals d'aquestes etapes:

- **Càrrega del nucli:** Inclou des que es dóna l'ordre d'instanciar la VMI fins que el nucli ha acabat d'inicialitzar el maquinari bàsic, moment en el que transfereix el control a l'initrd.
- **Arrancada de l'initrd:** Càrrega del mòduls i controladors necessaris per a poder accedir als dispositius i muntar els sistemes de fitxers que contenen el sistema operatiu instal·lat. Una vegada muntat el sistema de fitxers, se li transfereix el control al procés init.
- **Seqüència rcN.d:** El procés *init* executa tot el conjunt de seqüències d'inicialització de les aplicacions i serveis que es troben dins l'estructura d'arrancada *System V*. Entre elles es troba la seqüència de codi injectat per la personalització del paràmetre *init*, que és la darrera en executar-se, i de la que depenen la resta d'etapes.
- **Descàrrega del LQM:** Aquesta és la primera etapa que ja correspon estrictament al procés d'implantació, i inclou majoritàriament el temps destinat a la descàrrega dels fitxers necessaris per a instal·lar i configurar el gestor de cues. En aquest cas en concret, es descarrega un paquet Debian que conté tots els fitxers necessaris i les instruccions per dur a terme la instal·lació del gestor de cues.
- **Configuració del LQM:** Abans d'instal·lar, cal preparar el sistema preestablint alguns valors de configuració necessaris per a la instal·lació del gestor de cues. En el cas

particular que es ocupa, implica configurar algunes variables del gestor de paquets mitjançant el sistema de configuració *debconf*, propi de Debian.

- **Instal·lació del LQM:** Una vegada preconfigurat el gestor de cues, ja es pot desempaquetar i instal·lar a la seva ubicació respectiva. Aquesta etapa és que que es demora més temps, ja que inclou la descompressió del paquet Debian del gestor de cues. En concret, s'ha implantat Condor, que en particular mida uns 74 MiB.
- **Resolució de dependències:** Abans de poder executar el gestor de cues cal assegurar-se que el sistema disposa de totes les seves dependències de programari correctament instal·lades i configurades. El mecanisme de gestió de dependències entre paquets Debian és molt sofisticat, i existeixen eines automatitzades que faciliten en gran mesura la resolució d'aquestes dependències. Aquesta etapa inclou, des de l'anàlisi del graf de dependències del gestor de cues, fins a la descàrrega i configuració d'aquestes dependències necessàries per al gestor de cues.
- **Arrancada del LQM:** Finalment, una vegada instal·lats i configurats el gestor de cues, així com les seves dependències, el servei ja es pot engegar. Amb la finalització d'aquesta etapa es conclou l'acció de la implantació, i es cedeix el control de la màquina a l'usuari.

Etapa	Mínima (s)	Mitjana (s)	Màxima (s)	Desviació mitjana (s)
Càrrega del nucli	8	8'6	9	0'48
Arrancada de l'initrd	2	2'6	3	0'48
Seqüència rcN.d	11	12	13	0'4
Descàrrega del LQM	6	6'6	9	0'96
Configuració del LQM	5	6'6	7	0'64
Instal·lació del LQM	20	28'2	32	3'28
Resolució de dependències	12	14'6	20	2'16
Arrancada del LQM	0	0'2	1	0'32
Temps total d'arrancada	81	82'2	84	0'72

Taula 1: Durada de les etapes del mètode init

A la taula 1 es poden observar els valors de la durada mínima, mitjana i màxima de cada etapa, calculats a partir de les mesures preses en les diferents execucions. També s'inclou una

fila addicional amb els valors estadístics del temps total d'arrancada. Cal notar que aquesta darrera fila no es correspon al resultat de sumar els valors de les files anteriors. Això és degut a que en totes les simulacions efectuades, les diverses etapes de la implantació equilibraven els seus temps d'execució, de manera que una etapa ràpida venia seguida d'una que tardava més que la mitjana.

A la figura 11 es pot observar amb més detall la contribució de cada etapa al total de temps emprat per l'arrancada i implantació de la VMI.

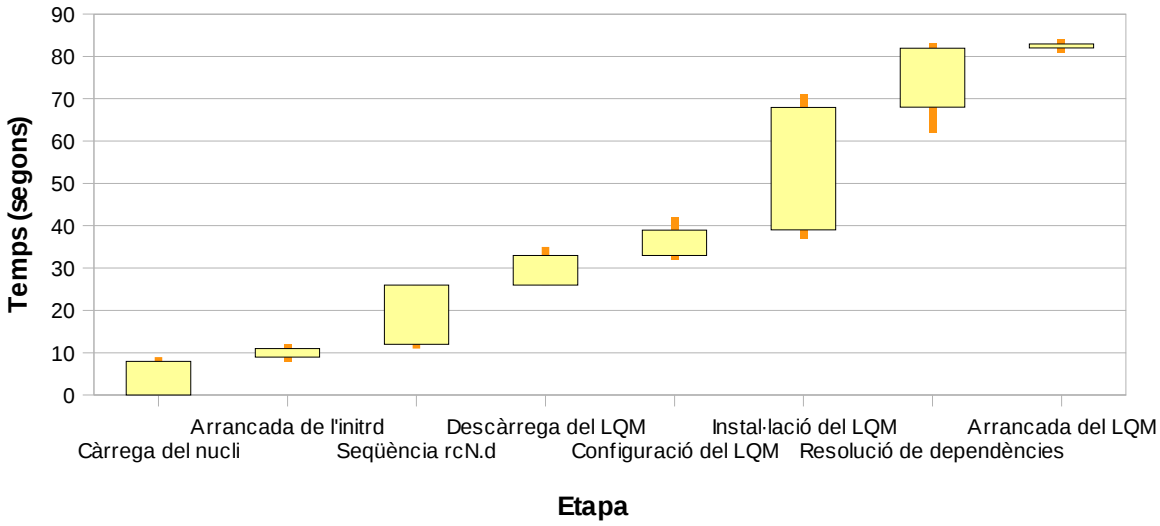

Figura 11: Seqüència temporal del mètode init

En totes les execucions que s'han dut a terme, el procés d'arrancada i implantació ha tardat entre 83 i 84 segons. D'aquest temps, uns 26 segons es corresponen a les tres primeres etapes d'arrancada del sistema, i els 57 restants a les altres cinc etapes destinades a la implantació. D'aquesta manera el procés d'implantació representa un 68 % del temps total d'arrancada de la màquina.

Cal comentar que una vegada implantat i arrancat el gestor de cues, aquesta màquina virtual no està disponible immediatament per a l'execució de les tasques enviades pel seu usuari, ja que existeix un cert retard entre el moment en que s'arranca el gestor de cues, i el node d'execució corresponent a aquesta màquina apareix com a disponible al gestor de cues central. En la gran majoria de les execucions que s'han dut a terme per a aquest experiment, aquest retard ha estat inferior als cinc segons, però depenent de la configuració, aquest retard pot arribar al minut. Per altra banda, hi ha altres retards que es poden afegir a aquest i que

poden endarrerir encara més l'execució de les tasques sobre les VMI, com el procés de negociació o la pròpia transferència de la tasca al node d'execució, tot i que aquests aspectes ja queden fora de l'àmbit d'estudi d'aquest experiment.

Finalment, i una vegada analitzats els resultats dels experiments, podem concloure que mitjançant aquest experiment s'ha pogut demostrar la viabilitat i efectivitat d'aquesta tècnica, ja que s'ha aconseguit dur a terme amb èxit la implantació de la VMI en totes les repeticions efectuades, sense afectar en cap altre punt el funcionament normal del sistema operatiu instal·lat, i amb un consum de temps molt tolerable.

Possibles extensions i millores

En la recerca que s'ha dut a terme per a desenvolupar aquesta tècnica ja s'han anat resolent i refinant la gran majoria de limitacions que s'han anat descobrint. Tot i així, encara resten alguns aspectes que es podrien millorar per tal de simplificar i flexibilitzar el procés d'implantació. En les proves que s'han fet, sempre s'ha treballat amb nuclis que requerien d'una imatge en RAM inicial per a poder arrancar. En aquestes condicions els paràmetres del nucli, i molt concretament el paràmetre *init*, és avaluat per aquesta imatge de RAM inicial. En el cas que el nucli no requerís de cap imatge en RAM per engegar (per ja dur encastats tots els mòduls necessaris), el responsable d'avaluar els paràmetres seria el mateix nucli. En aquest cas, és possible que la funció que avalua els paràmetres al nucli els analitzi de manera diferent a com ho fa la imatge en RAM i, per tant, podria aparèixer alguna problemàtica addicional, com altres caràcters que necessitin ser escapats. En qualsevol cas, no s'espera que aparegui cap complicació d'especial rellevància.

4.2.2. Mètode chroot

El segon mecanisme d'implantació planteja un enfocament molt més tradicional a la tasca d'assegurar un entorn d'execució adequat dins les VMI. En aquesta segona tècnica, es requereix un sistema hoste que munti els sistemes de fitxers adients de la VMI per a poder-hi injectar l'entorn d'execució. Aquest accés als sistemes de fitxers de la màquina virtual es duu a terme mitjançant l'ús de la tècnica de *chrooting*.

Una vegada es té accés als sistemes de fitxers, es poden modificar a voluntat els continguts de la màquina virtual com si aquesta s'estigués executant, de manera que aconseguir l'objectiu de la implantació esdevé relativament senzill.

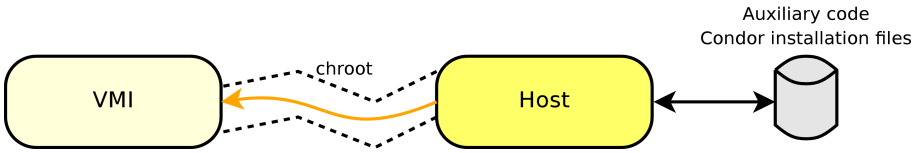

Figura 12: Components del mètode chroot

Funcionament

Així doncs, aquesta tècnica d'implantació rau en la possibilitat de poder manipular directament els continguts dels sistemes de fitxers de la VMI. Existeixen multitud de combinacions possibles pel que fa a la configuració dels sistemes de fitxers, i aquesta tècnica hauria de ser capaç de detectar i de treballar en tots els escenaris possibles.

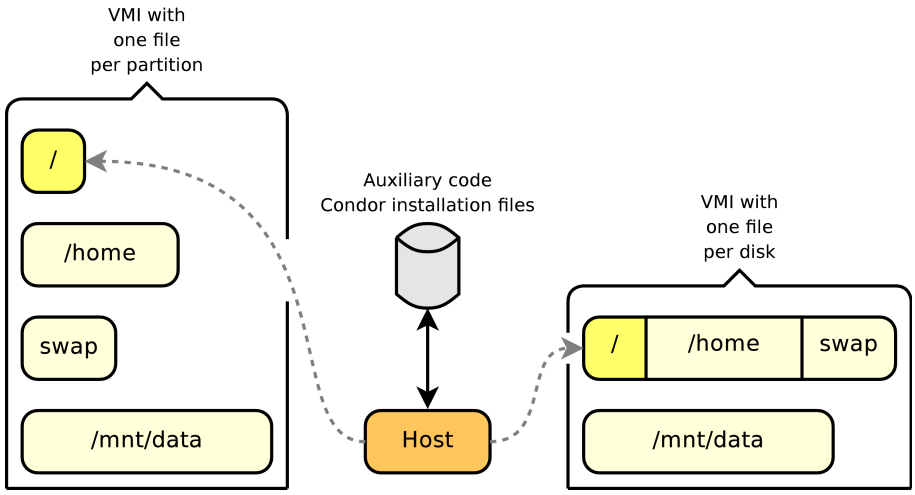

Figura 13: Configuració dels sistemes de fitxers

Cada VMI està formada per un o més sistemes de fitxers, que contenen el sistema operatiu, llibreries i tota la resta de programari instal·lat. Aquests sistemes de fitxers s'especifiquen en arxius imatge, i aquests poden respondre a diverses estructures, algunes de les quals apareixen al diagrama anterior.

- **Una imatge per cada sistema de fitxers:** En aquesta configuració, cada sistema de fitxers s'especifica en un arxiu per separat, de manera que una VMI amb tres sistemes de fitxers diferents estarà formada per almenys tres arxius d'imatge.

- **Una imatge per disc o volum:** En aquesta configuració, un conjunt determinat de sistemes de fitxers es troben agrupats dins una estructura lògica de nivell superior. En la majoria dels casos, aquesta estructura ressembla un disc amb una taula de particions, de manera que mitjançant aquesta taula de particions es poden delimitar els diferents sistemes de fitxers que hi ha continguts dins la imatge.

  En altres casos, però, els diferents sistemes de fitxers es poden agrupar dins una estructura més complexa, per exemple davall un gestor de volums lògics com LVM o similar. En aquest projecte, emperò, s'ha obviat aquesta darrera casuística per tal de reduir l'àmbit de la recerca i de les proves.

L'anàlisi de la configuració de les VMI es duu a terme emprant l'eina *kpartx*, la qual permet detectar l'estructura de les particions existents en un arxiu d'imatge i muntar-les de manera individual. Per poder dur a terme la implantació amb èxit, és necessari reconstruir l'arbre de directoris de la VMI de manera anàloga a l'estructura que tendria si estigués en execució, tal com es descriu al fitxer de configuració */etc/fstab*. Aquest objectiu es pot dur a terme imitant el procés d'arrancada del nucli, pel que fa a les tasques de muntatge dels sistemes de fitxers:

- En primer lloc cal muntar el sistema de fitxers arrel (/), que vendrà indicat en els paràmetres del la VMI. A partir d'aquest directori arrel en penjaran la resta de sistemes de fitxers.
- Una vegada muntada l'arrel ja es pot accedir al fitxer */etc/fstab*, ja que el directori /etc no pot estar en un sistema de fitxers diferent de l'arrel. Aquest fitxer és el responsable d'indicar quins sistemes de fitxers formen part de la màquina virtual, i en quina branca de l'arbre de directoris s'han de muntar.
- A continuació, amb la informació extreta del fitxer */etc/fstab* es poden anar muntant la resta de sistemes de fitxers que formen part de la VMI en les seves rutes respectives de l'arbre de directoris.

Fet això, ja disposam d'un arbre de directoris que ressembla quasi en la seva totalitat l'estructura que tendria si la màquina virtual es trobés en execució. Per a poder acabar de simular aquest entorn d'execució cal muntar també alguns sistemes de fitxers virtuals, que proporcionen serveis auxiliars i que poden resultar imprescindibles per a executar algunes

aplicacions. En particular, caldrà muntar els sistemes de fitxers */dev* i */proc*, on el primer proporciona informació sobre els diferents dispositius presents a la màquina [E26], i el segon proporciona informació sobre el maquinari i els processos que es troben en execució [E27]. El mecanisme per muntar aquests dos sistemes de fitxers és lleugerament diferent, ja que és suficient «clonar» l'estructura homòloga del sistema hoste a la branca concreta de l'arbre de la màquina virtual, el que s'anomena un muntatge lligat. Finalment, una vegada reconstruït l'arbre de directoris de la màquina virtual ja es pot fer ús de la tècnica de *chrooting* per a «entrar» dins la màquina virtual i executar-hi aplicacions com si aquesta s'estigués executant.

En aquest segon mètode d'implantació s'ha intentat reaprofitar al màxima la recerca i la feina feta en el primer mètode. En particular, la seqüència de comandes que s'executa dins la màquina virtual mitjançant *chrooting* és la mateixa que la del primer mecanisme. D'aquesta manera se'n reaprofita el codi i es disminueixen els costos de manteniment. La funció d'aquesta seqüència de comandaments, com ja s'ha descrit al mecanisme d'implantació anterior, és verificar que la màquina virtual disposi d'un entorn d'execució propici per a les tasques que enviï l'usuari o, en cas contrari, dur a terme les accions necessàries per a implantar aquest entorn i assegurar que compleixi totes les condicions necessàries.

Avantatges i inconvenients

Aquesta tècnica d'implantació no exigeix tants requeriments com el mecanisme de manipulació del paràmetre *init* i, per tant, és aplicable a més escenaris. A continuació es detallen els principals avantatges i inconvenients d'aquest mecanisme.

- ✔ **Compatible amb la totalitat de VMI:** Aquesta tècnica pot aplicar-se indistintament tant a aquelles màquines virtuals que duen el nucli de Linux encastat, com en aquelles on cal proporcionar-lo externament, ja que no cal arrancar la màquina per a fer la implantació. D'aquesta manera, mitjançant aquesta tècnica és possible implantar qualsevol tipus de màquina virtual, tot i que la complexitat del mètode (en particular del muntatge de l'arbre de directoris) por variar segons la tipologia a la qual pertanyi.
- ✔ **Consum nul d'amplada de banda:** El fet que la implantació s'hagi de dur a terme en local implica que els fitxers necessaris per la implantació poden estar emmagatzemats al mateix sistema hoste que duu a terme la operació. D'aquesta manera, i en contraposició als requeriments lineals d'amplada de banda del mecanisme init,aquest

mètode pot dur a terme la implantació sense haver de descarregar cap gran volum de dades de la xarxa.

- ✔ **Reutilització de la feina:** La seqüència de comandaments que s'utilitza per a dur a terme la implantació de l'entorn d'execució és la mateixa que la que s'ha desenvolupat i provat per al mecanisme init. Així doncs, aquest mecanisme ens permet de reaprofitar tota la recerca i la tasca feta i establir una única base de codi a mantenir i millorar, el que permet optimitzar els costos.
- ✘ **Anàlisi de la configuració dels sistemes de fitxers:** Com ja s'ha detallat al punt de tipologia, les possibles combinacions en la configuració dels sistemes de fitxers constitueixen un dels punts febles d'aquest mecanisme, ja que serà necessari analitzar i detectar l'estructura, tipus i ruta de muntatge de tots els sistemes de fitxers per tal de poder reconstruir amb l'arbre de directoris de la VMI. La construcció amb èxit d'aquest arbre de directoris és la peça fonamental d'aquest mecanisme i és imprescindible per a poder dur a terme la implantació.
- ✘ **Preprocessament previ:** Com que cal un sistema hoste per a construir l'arbre de directoris de la VMI, aquesta tècnica d'implantació requereix que es dugui a terme localment i de manera prèvia a la instanciació. Això comporta que les VMI que envia l'usuari han de ser preprocessades abans no es tengui la certesa que seran aptes per a l'execució de les tasques.
- ✘ **Crides de sistema:** Finalment, un dels aspectes en els que l'abstracció proporcionada per la tècnica de *chrooting* presenta alguna deficiència respecte de la proveïda per la virtualització és en les crides a sistema. En una VMI virtualitzada, les crides a funcions del sistema les resol el nucli de la VMI o, en el seu defecte, una instància del nucli proporcionada externament i que és exclusiva per a aquella instància de la VMI. Però en un entorn *chroot*, les crides a sistema les resol el nucli del sistema hoste, que depenent del tipus de crida pot retornar una resposta que no es correspongui amb el context de la VMI.

  Aquesta característica dels entorns *chroot* té un doble efecte. Per una banda proporciona a la VMI d'un entorn amb el maquinari configurat i en funcionament com, per exemple, els sistemes de fitxers o la xarxa. Però per altra banda algunes crides a sistema retornen valors erronis, com les que determinen el nom de la màquina actual

(hostname), que retornen incorrectament el valor corresponent a la màquina hoste, en comptes del nom configurat a la VMI.

Proves de rendiment

De manera anàloga al primer mecanisme, s'ha dissenyat un conjunt de proves per tal de comprovar el bon funcionament d'aquesta tècnica i per mesurar-ne el seu rendiment. El joc de proves és molt similar a l'emprat en el mecanisme *init* per tal d'afavorir una millor comparació entre els resultats respectius. Seguint el mateix sistema que per al mecanisme d'implantació anterior, les proves s'han dut a terme sota el mateix maquinari ja descrit, realitzat també una dotzena d'execucions i respectant les condicions inicials mitjançant el buidatge de les memòries caus.

En aquest cas, a l'hora de mesurar el rendiment d'aquest mecanisme, el procés d'implantació s'ha dividit en set etapes, de les quals quatre són comunes amb el mecanisme *init*. Aquestes quatre etapes compartides són la descàrrega, instal·lació i configuració del gestor de cues, i la seva funció ja s'ha descrit a l'apartat anàleg del mecanisme anterior. A continuació es descriuen amb més detall les característiques i funcions principals de les tres etapes que són exclusives d'aquest mecanisme:

- **Muntar directoris:** Inclou l'anàlisi de la configuració dels sistemes de fitxers de la VMI i la posterior reconstrucció de l'arbre de directoris necessari per a simular-hi l'entorn d'execució de la màquina virtual.
- **Injecció del codi:** En aquesta etapa es copia el codi necessari per a fer la implantació de la VMI (recordem que és el mateix codi que s'utilitza per al mecanisme *init*) i a continuació s'executa mitjançant la tècnica del *chroot*.
- **Desmuntar directoris:** Una vegada duta a terme la implantació del gestor local de cues, és necessari desmuntar de manera adient els sistemes de fitxers que componen l'arbre de directoris per assegurar que la informació queda desada de manera consistent i es preservarà en cas de fallada inesperada del maquinari.

A la taula 2 es poden observar els valors de la durada mínima, mitjana i màxima de cada etapa, calculats a partir de les mesures preses en les diferents execucions. També s'inclou una fila addicional amb els valors estadístics del temps total d'arrancada.

Etapa	Mínima (s)	Mitjana (s)	Màxima (s)	Desviació mitjana (s)
Muntar directoris	0	0,6	1	0,48
Injecció del codi	2	2,4	3	0,48
Descàrrega del LQM	5	5,4	6	0,48
Configuració del LQM	7	7,8	9	0,64
Instal·lació del LQM	25	29	34	3,2
Resolució de dependències	3	3,2	4	0,32
Desmuntar directoris	0	1	2	0,4
Temps total d'arrancada	45	49,4	54	3,28

Taula 2: Durada de les etapes del mètode chroot

A la figura 14 es pot observar amb més detall la contribució de cada etapa al total de temps emprat per a la implantació de la VMI.

En totes les execucions que s'han dut a terme, el procés d'implantació ha tardat entre 42 i 54 segons, on uns 42 segons es corresponen a les tres etapes compartides amb el mètode d'implantació init. D'aquesta manera, l'ús d'aquest mètode en els supòsits en que és possible, pot suposar una reducció del temps d'implantació de més del 40%.

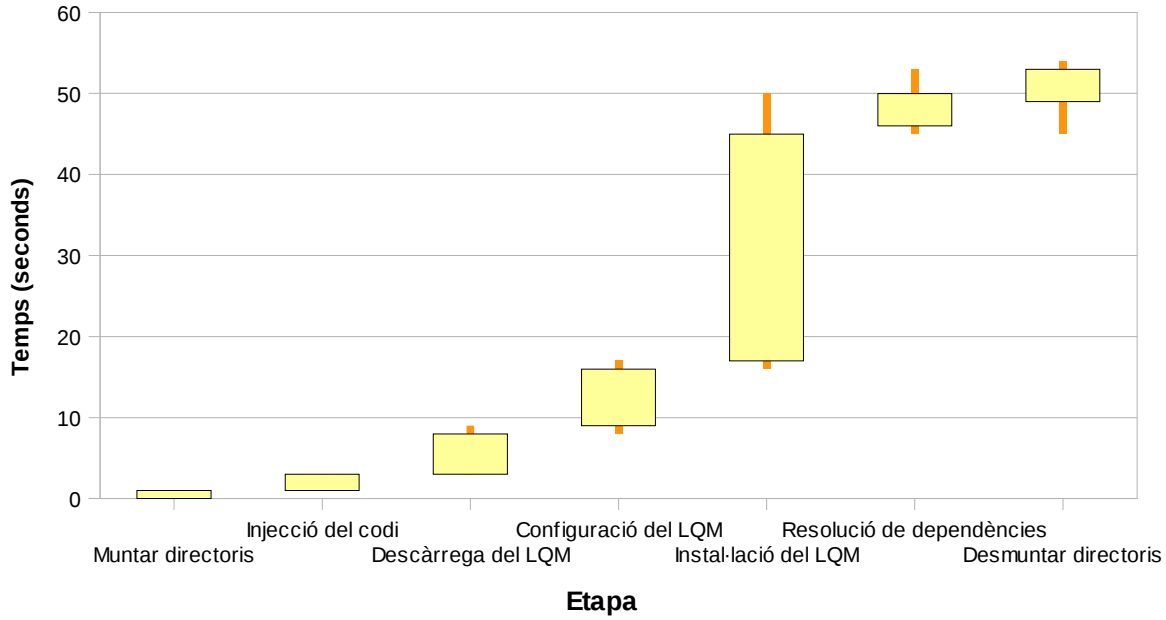

Figura 14: Seqüència temporal del mètode chroot

Una vegada dut a terme aquest procés d'implantació sobre una VMI, aquesta queda ja preparada per a ser instanciada en qualsevol altre ubicació, on apareixerà com a nou node d'execució.

Finalment, i com a comentari conclusiu, ressaltar que els resultats d'aquests experiments permeten demostrar la viabilitat i efectivitat d'aquesta tècnica, ja que s'ha aconseguit instal·lar el gestor de cues en totes les repeticions executades sense modificar el normal funcionament de la VMI, i amb una millora de temps respecte al mètode *init*.

Possibles extensions i millores

En aquest mecanisme d'implantació, les tècniques emprades són molt senzilles, conegudes, i àmpliament provades, pel que hi ha poques àrees susceptibles de millora. En tot cas, sí es podria ampliar el procés d'anàlisi, que actualment es fa de manera parcialment manual, per tal que pugui tractar qualsevol tipus de VMI de cara al procés d'implantació de manera completament automàtica, simplement subministrant el fitxer de descripció de la VMI.

4.3. Aturada de les màquines virtuals

Al principi d'aquest capítol s'ha esmentat el cicle de vida de les imatges virtuals i, a l'apartat anterior, s'ha tractat la primera fase: la implantació i ulterior arrancada de les màquines virtuals. Quan les màquines virtuals d'un usuari estan arrancades, s'organitzen automàticament en un cloud privat que constitueix l'entorn d'execució de les tasques que ha enviat. L'objectiu d'aquest apartat és estudiar el procés d'aturada de les màquines virtuals, segons l'esquema adjunt de la figura 15:

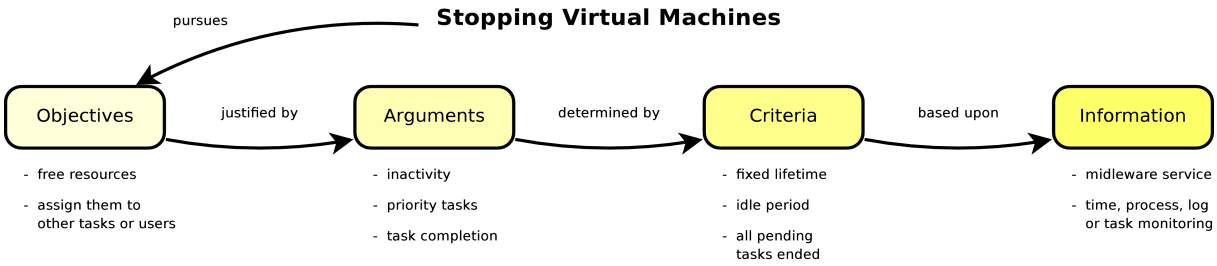

Figura 15: Flux de decisió per aturar les màquines virtuals

L'aturada de les màquines virtuals persegueix un objectiu ben concret i definit: l'entorn d'execució format per aquestes màquines virtuals consumeix un conjunt de recursos que estan en continua disputa i, per tant, cal establir algun mecanisme de control que asseguri que un conjunt de màquines no monopolitza l'ús d'aquests recursos. L'objectiu d'aturar les màquines virtuals és, doncs, alliberar els recursos que s'estiguin emprant per a poder destinar-los a altres tasques o usuaris.

Per altra banda, no cal perdre de vista que l'objectiu d'aquest projecte és executar tasques. En particular, aquest projecte es basa en els entorns grids i en les tecnologies de virtualització per a definir una arquitectura on els usuaris puguin executar les seves tasques en un entorn virtual dedicat. Per tant, és clau determinar el moment idoni per a apagar un conjunt de màquines virtuals, ja que si s'apaguen a destemps, és possible que l'execució d'aquestes tasques es vegi afectada. En termes generals, els motius que justifiquen la presa de la decisió d'aturar un conjunt de maquines virtuals, o cloud, d'un usuari són els següents:

- **Una vegada les tasques destinades a executar-s'hi s'han acabat amb èxit.** Aquesta condició és difícil de determinar si no es disposa d'un coneixement complet sobre l'estat de les màquines i dels gestors de cues involucrats. Per altra banda aquest motiu, per si sol, podria esdevenir insuficient a l'hora determinar el millor moment per a dur a terme l'aturada, i caldrà complementar-lo amb el següent.
- **No es preveu que arribin noves tasques que puguin executar-se en aquestes màquines.** És a dir, que les màquines virtuals que es pretenen apagar no siguin compatibles com a plataforma d'execució de noves tasques que puguin arribar en un futur proper. Aquesta condició serà complexa de provar, ja que en la gran majoria de casos no es disposarà d'una base de coneixement suficientment sòlida per a poder dur a terme aquestes previsions. En qualsevol cas, i com es veurà una mica més endavant, aquesta condició es pot aproximar, simplificant en gran mesura la implementació d'aquesta condició.

  Combinant aquesta segona condició amb la primera, ja es disposen de suficients arguments per a dur a terme l'aturada de les màquines virtuals, amb la seguretat que l'objectiu principal, és a dir, l'execució de les tasques dels usuaris, no es veurà afectat.
- **L'arribada de tasques amb major prioritat:** Aquesta condició, al contrari que les dues anteriors, és autosuficient. L'arribada al gestor de cues d'una o més tasques amb

una prioritat major que la de les tasques que en aquests moments ocupen els recursos ja constitueix argument suficient per a aturar les màquines i cedir els seus recursos a les tasques prioritàries.

L'únic però que cal comentar a aquesta condició és la possibilitat que l'aturada de les màquines interrompi treballs en execució. Per a aquests casos, en el proper apartat s'estudiaran les possibilitats de la suspensió i/o migració de les tasques.

Aquests són, en resum, els tres motius que permeten justificar l'acció d'aturar una màquina virtual. D'aquests cal derivar-ne un conjunt de criteris que permetin determinar quan concorren o no algun d'aquests motius. Aquests criteris basaran el seu procés de decisió en el conjunt d'informació d'estat disponible sobre les màquines virtuals. Per tant, depenent de la tècnica d'obtenció d'aquesta informació d'estat de les màquines virtuals, serà factible o no la determinació de certs criteris. En alguns casos, la informació proporcionada no serà suficient per a poder avaluar de manera exacta si concorre un determinat motiu i caldrà dur a terme alguna aproximació que en simplifiqui la implementació.

Els criteris derivats dels motius anteriors, que en suposen una implementació particular, són els següents:

- **Un límit de temps definit:** Es pot planificar l'aturada de les màquines una vegada transcorregut un període de temps determinat, que es pot fixar amb anterioritat depenent de les característiques de la màquina i dels treballs que s'hauran d'executar. L'avantatge d'aquest criteri és que no requereix cap tipus d'informació d'estat sobre la màquina o màquines virtuals que es volen apagar: és suficient conèixer l'instant en que foren enceses per determinar el moment en el que caldrà aturar-les. Per altra banda, és un criteri molt inexacte, ja que no permet assegurar que quan es dugui a terme l'aturada no hi hagi tasques executant-s'hi.
- **Un període d'inactivitat:** Un dels criteris més senzills és aturar una màquina virtual una vegada ha transcorregut un període d'inactivitat d'una durada determinada. D'aquesta manera es pretén assegurar que les tasques encomanades ja s'han acabat d'executar (el sistema resta inactiu) i el període d'inactivitat posterior serveix de marge durant el qual el sistema romandrà encara a l'espera de l'arribada de noves tasques compatibles, per així poder *reciclar* aquestes instàncies. L'addició d'aquest marge de

temps pretén reduir els efectes negatius que pugui causar una predicció incorrecte sobre l'arribada de tasques compatibles.

- **La compleció dels treballs:** En aquest criteri, les màquines virtuals d'un usuari s'apagaran quan els treballs que les requerien com a plataforma d'execució ja s'hagin completat. És el millor criteri de tots, ja que és el que redueix al mínim els possibles efectes adversos sobre l'execució de les tasques. L'únic aspecte que no té en compte és la previsió d'arribada de tasques que sol·licitin una màquina compatible i que puguin aprofitar algunes de les instàncies en execució. En aquest casos, si es vol minimitzar l'ocurrència d'aquesta situació, es pot combinar aquest criteri amb l'anterior, afegint un cert marge de temps d'inactivitat, per tal d'evitar haver de rellançar les màquines si arriben treballs compatibles.

Aquests són els criteris que permeten establir el moment idoni per a dur a terme una aturada controlada de les màquines virtuals i, en tots els casos, aquesta decisió es basa en el conjunt d'informació d'estat disponible de les màquines. És necessari, per tant, establir un o més mecanismes per tal d'extreure i de proporcionar aquesta informació que possibilita la presa d'aquesta decisió. Per altra banda, els criteris que es podran aplicar en un moment donat dependran de la qualitat de la informació disponible i, aquesta, indirectament del grau de seguiment que es faci de l'estat de les màquines. Per tant, els criteris que es poden aplicar en un moment donat queden lligats al mecanisme concret de monitorització que es faci servir.

Així doncs, els mecanismes de seguiment són els encarregats d'extreure i proporcionar la informació necessària per a poder determinar el millor moment per a dur a terme l'aturada de les màquines que conformen el cloud de l'usuari. Existeixen diverses tècniques o mecanismes de seguiment que, en una primera classificació, es poden dividir entre mecanismes de seguiment intern i mecanismes de seguiment extern.

Mecanismes de seguiment externs

Els mecanismes de seguiment externs són aquells que duen a terme la monitorització de manera externa a la màquina virtual que supervisen. En altres paraules, el procés que duu a terme aquest control s'executa en una màquina diferent a la supervisada. En aquest grup de mecanismes trobam dues tècniques de seguiment, segons la qualitat de la informació que són capaces d'obtenir:

- **Encapsulat dels treballs:** Aquesta tècnica consisteix en disposar d'algun mecanisme extern que exerceixi d'intermediari entre l'usuari i el gestor de cues que ha de llançar les màquines virtuals i els treballs. Cal recordar que les funcions d'aquest intermediari ja estan previstes a l'arquitectura descrita al capítol 3; aquest procés intermediari serà l'encarregat de llançar tant les màquines virtuals com els treballs que han d'executar-s'hi i, per tant, disposarà d'informació completa sobre el seu estat.

  Aquest mecanisme assoleix la màxima qualitat possible de la informació, ja que al interactuar directament amb els gestors de cues disposa d'informació privilegiada: coneix perfectament quines màquines virtuals hi ha instanciades en tot moment, quins treballs s'estan executant (i en quina màquina), els que estan encuats, així com també aquells que, tot i encara no estar encuats, ho estaran en un futur molt proper. En aquesta darrera casuística trobam els treballs als quals s'estan preparant les màquines virtuals, o dels quals encara se n'estan transmetent les dades de les VMI.

- **Control del gestor de cues:** A falta d'un procés intermediari, es pot intentar aproximar la informació mitjançant el seguiment de la informació retornada per les eines de control del gestor de cues. D'aquesta manera es pot determinar quines màquines virtuals i treballs s'executen en un moment donat, així com els treballs encuats. Aquesta tècnica, però, no pot disposar d'informació privilegiada, com seria els treballs que estan a punt d'arribar, o que estan sent processats per l'intermediari.

Mecanismes de seguiment interns

Els mecanismes de seguiment intern són aquells que obtenen la informació d'estat directament des de l'interior de les pròpies màquines que monitoritzen. El seu funcionament es basa en el seguiment d'un conjunt de variables o fonts d'informació, a partir de les quals els diferents criteris poden fer les seves conjectures. Entre les fonts d'informació disponibles trobam les següents:

- **Processos:** A partir de la taula de processos del sistema operatiu es pot determinar si hi ha treballs en execució, simplement observant la presència dels processos llançadora del gestor de cues. En cas que no apareguin, es pot assegurar que la màquina es troba desocupada, però no es pot extreure cap informació sobre la probabilitat d'ús futur d'aquesta màquina en treballs compatibles.

- **Càrrega del sistema:** El nivell de càrrega del sistema mesura el nombre mig de processos que competeixen per l'ús de la CPU en un interval de temps determinat i permet establir i delimitar els períodes d'inactivitat d'una màquina. Així doncs, segons el patrons de la càrrega del sistema s'intentarà inferir si s'estan executant treballs en una màquina concreta o si es troba desocupada. Tot i així, un període d'inactivitat no assegura l'absència de tasques d'usuari, ja que aquestes es podrien trobar bloquejades en operacions d'entrada/sortida i, per tant, podrien no veure's reflectides en la mesura de la càrrega del sistema.
- **Registres de bitàcola (*logs*):** Com ja s'ha descrit a l'apartat anterior, en cada màquina virtual s'estarà executant un gestor de cues local, que serà l'encarregat d'arbitrar l'execució de les tasques de l'usuari. Aquests gestors de cues solen utilitzar fitxers de bitàcola per a registrar les operacions que van duent a terme. Aquest mecanisme de seguiment proposa l'anàlisi d'aquests registres per a poder determinar, d'una manera més fiable que amb els dos mecanismes anteriors, els moments o intervals en els que una màquina està desocupada i pot ser aturada.

Actualment, i segons l'arquitectura definida al capítol 3, el procés intermediari encara no està implementat i, per tant, s'haurà de prescindir d'ell com a mecanisme d'obtenció d'informació d'estat sobre les màquines virtuals. Dels mecanismes de seguiment restants, s'han dedicat més esforços als externs, ja que afavoreixen una major descentralització en l'obtenció de la informació i una millor escalabilitat, factors clau en els entorns distribuïts en els que es mou aquest projecte.

4.3.1. Mètode del sentinella

En aquest projecte, la recerca i els experiments que s'han dut a terme per a comprovar la viabilitat dels mecanismes de seguiment i decisió a l'hora d'aturar les màquines virtuals s'han centrat en l'anàlisi dels registres i, en particular, dels registres de bitàcola dels gestors de cues locals. En aquest punt es detalla la implementació en forma de prova de concepte d'un senzill mecanisme que permet determinar el millor moment per dur a terme aquesta aturada programada. Les característiques principals d'aquest mecanisme, anomenat mecanisme del sentinella, són les següents:

- Utilitza una tècnica de seguiment interna, ja que el procés d'extracció de la informació d'estat de la màquina virtual s'hi executa a dins mateix. Per una banda, això el dota d'un accés més proper a les fonts d'informació, mentre que per l'altra s'assegura una gran escalabilitat del sistema.
- L'objectiu d'aquest mecanisme és poder determinar amb exactitud la presència de tasques en execució a la màquina supervisada, així com detectar els períodes d'inactivitat. L'absència de tasques en execució seguida d'un període d'inactivitat és un dels criteris que permet aplicar amb seguretat l'acció d'aturada d'una màquina virtual.
- No utilitza cap tipus de servei centralitzat que emmagatzemi la informació d'estat. La informació recollida no es transmet a l'exterior de la màquina virtual supervisada. Aquest mecanisme és capaç de prendre les decisions sobre l'aturada de la màquina sense cap tipus de coneixement global o procés de negociació extern.

Per a que aquest mecanisme funcioni, cal integrar-lo dins l'entorn d'execució del gestor local de cues durant l'etapa d'implantació. D'aquesta manera, quan la màquina arranqui, ja estarà preparat per a avaluar i prendre la decisió sobre el moment idoni de la seva aturada.

Funcionament

El gestor de cues que serveix com a entorn d'execució de les tasques de l'usuari s'ha configurat (durant la implantació) per a generar un conjunt de fitxers de bitàcola que registren l'estat del gestor de cues i les diferents operacions que va realitzant. Les diferents fonts d'informació utilitzades en aquest mecanisme, així com el coneixement que se n'extreu són les següents:

- L'absència de certs fitxers de bitàcola indica que aquesta màquina encara no s'ha fet servir com a node d'execució. Això podria deure's a diversos motius:
  - La màquina just acaba d'arrancar i encara no ha rebut cap treball.
  - S'han instanciat més màquines que tasques i part de les màquines no s'utilitzen.
  - El procés de negociació no ha estimat oportú assignar tasques a aquesta màquina.
- El contingut dels fitxers de bitàcola permet determinar si hi ha tasques en execució o bé si ja han finalitzat. Aquesta informació es pot extreure de manera més ràpida i

senzilla comprovant a la taula de processos del sistema operatiu la presència dels processos auxiliars que donen suport a l'execució de les tasques.

- La data de modificació dels fitxers de bitàcola pot indicar el temps d'inactivitat d'aquest node d'execució.

La combinació d'aquestes tres fonts proporciona coneixement suficient per inferir la ocurrència d'una situació d'inactivitat perllongada en el temps i, així, justificar l'alliberament dels seus recursos mitjançant l'apagada de la màquina virtual.

Experimentació

Les proves realitzades han tengut com a objectiu comprovar la viabilitat d'aquest mecanisme i assegurar el seu correcte funcionament. L'experiment que es dugué a terme fou el següent:

- S'instanciaren tres màquines virtuals per a proporcionar tres espais d'execució virtuals (A, B i C), cadascun d'ells llest per a executar tasques.
- El mecanisme del sentinella instal·lat als tres espais d'execució virtuals, configurat per a apagar les màquines virtuals en detectar que no s'executen tasques i que han transcorregut quinze minuts d'inactivitat.
- A continuació es van enviar una dotzena de treballs d'una durada aproximada de dos minuts cadascun, que el gestor de cues va anar distribuint pels diferents espais d'execució.
- Transcorreguts menys de deu minuts, totes les tasques ja havien finalitzat i deixaren les tres màquines virtuals sense activitat. En aquest moment, el mecanisme del sentinella comença el compte enrere de quinze minuts.
- Abans d'haver transcorregut els quinze minuts, es va enviar una altra tasca, amb l'objectiu de comprovar si la màquina virtual que l'executaria reiniciaria el seu compte enrere.
- Aquesta tasca addicional va ser assignada pel gestor de cues a l'espai d'execució virtual A. De resultes d'aquesta execució, el mecanisme del sentinella detectà l'activitat i reinicià el compte enrere, situant-lo de nou en quinze minuts.

- Els sentinelles dels espais d'execució virtuals B i C completaren satisfactòriament l'aturada de les seves màquines, una vegada transcorreguts quinze minuts des de la finalització de l'execució de les primeres dotze tasques.
- L'espai d'execució virtual A, que el seu compte enrere havia estat reiniciat per l'execució de la tretzena tasca, va completar també satisfactòriament la seva aturada en complir-se quinze minuts des del final de la seva execució.

Els resultats d'aquest experiment permeten concloure que el mecanisme del sentinella és un mecanisme molt vàlid, tot i la seva senzillesa, per a poder determinar un bon moment per a dur a terme l'aturada.

4.4. Suspensió i migració de les màquines virtuals

Els administradors d'un entorn grid solen configurar els seus gestors de cues per a intentar maximitzar o bé la productivitat (mesurada segons el nombre de tasques completades per unitat de temps), o bé la ocupació mitjana de les màquines (mesurada a partir de paràmetres com la càrrega del sistema). Aquestes dues magnituds estan contraposades a la pràctica i suposen els dos punts de vista enfrontats respecte als models d'utilització de les màquines. Per una banda, els usuaris voldrien que les seves tasques finalitzessin el més prest possible -i, per tant, empressin tots els recursos disponibles-, mentre que els administradors intenten que no es desaprofitin recursos, maximitzant la utilització de les màquines. En aquesta disputa, generalment s'opta per un punt intermedi que afavoreix una magnitud sense perjudicar severament l'altre. D'aquesta manera s'intenta extreure el màxim rendiment de la infraestructura computacional de què es disposa i, a la vegada, oferir el millor servei possible als usuaris.

És obvi que la cancel·lació o interrupció d'un treball és un pas enrere en aquest direcció, ja que suposa perdre tota la feina feta fins en aquell moment i, depenent del tipus de tasca, també pot provocar retards en altres treballs que depenguin dels seus resultats. L'escenari d'haver de cancel·lar una tasca es pot minimitzar, però tot i així és inevitable. A l'apartat anterior s'ha parlat de les estratègies i mecanismes per dur a terme l'aturada de les màquines virtuals, posant esment en evitar, en la mesura del possible, la interrupció de tasques que estiguessin en execució. Amb el mecanisme del sentinella, només es provoca l'aturada d'aquelles màquines virtuals que no tenen cap treball en execució i que, a més, han romàs inactives durant un cert

període de temps. Emperò, existeix dues situacions en les quals no es pot evitar la cancel·lació d'un treball en curs:

- l'arribada de treballs amb una prioritat superior i que requereixen dels recursos que actualment s'estan utilitzant.
- si la màquina física no és una màquina dedicada en exclusiva a l'execució de treballs, aquesta execució no pot interferir amb l'usuari: en el moment en que l'usuari requereixi utilitzar la màquina, els treballs hauran d'alliberar els recursos en el menor temps possible.

En un sistema virtualitzat, la pròpia arquitectura -definida al capítol 2- proporciona nous mecanismes [18] que permeten minimitzar l'impacte que aquesta cancel·lació pot tenir sobre el rendiment del grid. Aquests mecanismes amplien el cicle de vida de les tasques, en la manera que es mostra a la figura 16, i cada un dels mecanismes ofereix una suspensió més *profunda* que l'anterior.

Suspensió

Una màquina virtual pot ser suspesa pel seu hipervisor i deixada en un estat de repòs o pausa en la qual no utilitzi la CPU, tot que continuarà resident en memòria. En aquest casos, si l'usuari o la nova tasca requereixen utilitzar aquest espai de memòria, el sistema operatiu pot forçar la paginació del procés d'aquesta màquina virtual a disc. La paginació d'un procés suposa bescanviar l'espai de memòria emprat a la memòria principal, per espai de memòria virtual que es sol desar al disc. D'aquesta manera, el procés que sustenta la màquina virtual no utilitza la memòria principal i resta a l'espera de poder reprendre la seva execució, a partir del seu estat desat a la memòria virtual. Aquesta paginació del procés a disc es pot assimilar amb l'estat d'hibernació -en el sentit que s'assoleix la mateixa alliberació dels recursos- però amb una diferència formal, ja que un procés suspès i paginat a disc encara consta a la llista de processos i pot reprendre l'execució més ràpidament.

Existeixen estudis [19] sobre els efectes de la suspensió de tasques en el rendiment global d'un grid. En particular, en aquest estudi s'analitza la influència de la suspensió preventiva de les tasques, a l'hora de millorar el temps de resposta mig de les tasques. A la documentació [E28] del gestor de cues es descriuen alguns mecanismes que permeten establir una política de suspensió flexible que intenta assolir una ocupació màxima de les màquines. Segons

aquesta política, cal dividir els treballs (les VMI) en dos tipus, els poden ser suspesos (baixa prioritat), i els que no (alta prioritat). Cada màquina física estaria configurada amb dos *slots* d'execució, cadascun destinat a un tipus diferent de VMI. El funcionament d'aquesta política seria el següent:

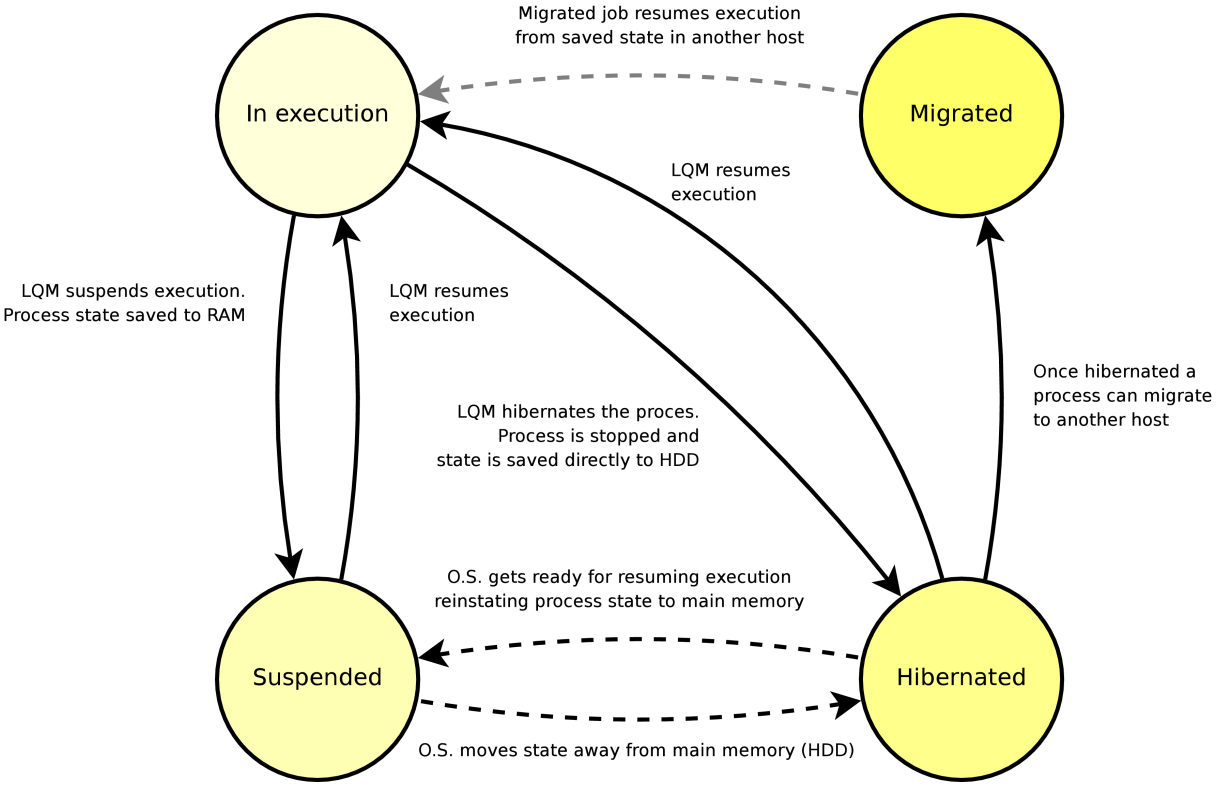

Figura 16: Diagrama d'estats de la suspensió, hibernació i migració

- Les VMI que poden ser suspeses podrien executar-se en qualsevol màquina que estès desocupada, ocupant el seu *slot* corresponent. L'arribada a la màquina d'una VMI d'altra prioritat, que començaria a executar-se a l'altre *slot*, provocaria que la VMI de baixa prioritat es suspengués i hagués d'esperar a que l'altra VMI acabi per reprendre la seva execució.
- Les VMI que no poden ser suspeses preferirien executar-se en màquines desocupades. Si no hi hagués cap màquina desocupada aquestes VMI podrien executar-se en una màquina on s'estès executant una VMI de baixa prioritat, provocant la seva suspensió, a la manera descrita al punt anterior.
- En tot moment, en una màquina física només es podria estar executant una VMI, independentment del seu tipus.

- En cap cas dues VMI del mateix tipus poden coincidir en una mateixa màquina física. En altres paraules, les VMI de baixa prioritat, que poden ser suspeses, no es poden provocar la suspensió les unes a les altres.

Amb aquesta política es persegueix reduir el temps de resposta de les tasques, evitant la cancel·lació de les tasques i substituint-la per la suspensió temporal de la seva execució. Per a més informació sobre aquesta estratègia, consultar la documentació del gestor de cues [E28].

Hibernació

Una altra de les funcionalitats que proporcionen els hipervisors és la possibilitat d'hibernar les instàncies de les màquines virtuals que gestionen. La hibernació també implica la suspensió de l'execució de la VMI però, en aquest cas, l'alliberament del seus recursos és immediata, ja que l'estat del procés es desa directament al disc i el procés es finalitza. La hibernació d'una VMI consta de les fases següents:

- En primer lloc, quan l'hipervisor recupera el control del processador, atura l'execució de la màquina virtual.
- A continuació desa el seu estat d'execució (registres i taules del processador, espai de memòria, descriptors de fitxers, entre d'altres) en un fitxer a disc.
- Finalment, evacua la màquina virtual, alliberant els recursos que estava emprant.

Aquest estat desat en disc és el que permet, de manera anàloga al cas de la suspensió, la represa de la màquina virtual en el mateix estat d'execució en la que estava just abans de la hibernació. El sistema gestor de cues disposa d'algunes variables de configuració [E29] que permeten alternar entre la suspensió i la hibernació, a l'hora de pausar l'execució d'una màquines virtual. Aquestes variables es poden configurar, o bé per l'administrador del sistema com a part de la política d'ús dels recursos, o bé per l'usuari segons les seves necessitats.

La hibernació suposa un pas previ imprescindible per a poder dur a terme el següent i darrer mecanisme, la migració de la VMI a un altre màquina.

Migració

Els dos mecanismes anteriors redueixen l'impacte de la cancel·lació d'una VMI substituint-la per una pausa temporal de la seva execució. En alguns casos, aquesta pausa pot

ser de curta durada -per exemple, per donar pas a l'usuari o a una tasca prioritària- o, per altra banda, la suspensió de l'execució pot ser per un període més llarg -per exemple, per una aturada de manteniment de la màquina, o l'arribada d'una tasca prioritària de llarga duració-.

El mecanisme de migració permet reduir encara més l'impacte, traslladant la VMI de la màquina on havia de ser cancel·lada a una altra. Existeixes múltiples estudis [20] sobre les tècniques utilitzades per a implementar la migració de les màquines virtuals, alguns [21] fins i tot en orientats a entorns distribuïts. L'objectiu d'aquest mecanisme és permetre l'alliberació dels recursos de còmput de la primera màquina, que són cedits a qui els requeria, mentre que la VMI afectada es reubicada en una altra màquina amb recursos lliures, o bé desada temporalment en un repositori a tal efecte, a l'espera que s'alliberin els recursos necessaris per a la seva execució.

Finalment, comentar que algunes d'aquestes funcionalitats, en particular, la hibernació i la migració, poden no estar presents en alguns entorns grid virtualitzats. La seva disponibilitat depèn, tant del conjunt de funcionalitats ofertes per l'hipervisor, com del seu suport per part del gestor de màquines virtuals. Per a les proves efectuades en aquest projecte, s'ha utilitzat Xen com a hipervisor i Condor com a gestor de cues. Mentre que Xen sí permet tant la suspensió, hibernació com la migració de les tasques, Condor encara no suporta la migració d'instàncies de màquines virtuals d'un node físic a un altre. Tot i així, sí que es pot emular aquest comportament enviant les VMI al gestor de cues encapsulades de manera adient, tal i com s'introdueix al punt 3.5 d'aquest article [22].

En tot cas, el sistema implementat proporciona una interfície mínima per tal de poder controlar les ordres de suspensió i d'hibernació, i està preparat per poder ampliar el suport a la migració si s'utilitza un altre gestor de cues, o bé quan aquesta funcionalitat sigui implementada a Condor.

5. Conclusions i línies obertes

En aquest treball s'ha presentat una arquitectura per a la gestió de recursos virtuals en entorns grid orientada a l'execució de tasques sobre màquines virtuals. Aquestes màquines virtuals implementen un entorn d'execució virtual que proporciona a les tasques que s'hi executen un conjunt d'avantatges i noves funcionalitats que permeten millorar el rendiment de les aplicacions i, per extensió, del grid en global.

S'ha dut a terme una anàlisi dels requeriments d'aquesta arquitectura i de les seves implicacions específiques a l'hora d'aplicar-la als entorns grid. S'ha posat una atenció especial a la compatibilitat cap endarrere, de manera que la seva aplicació no distorsioni l'execució dels treballs sobre els recursos físics tradicionals.

El desenvolupament i l'experimentació d'aquest projecte s'han centrat en aquells components de l'arquitectura que participen en el cicle de vida dels treballs i de les màquines virtuals, agrupats en un subsistema anomenat gestor d'espais d'execució virtual. Aquests espais d'execució virtual es generen a partir d'imatges de màquina virtual (VMI) que, o bé ja estan emmagatzemades al sistema, o bé les proporciona l'usuari annexades amb els treballs. Al primer apartat del capítol 4 s'han definit i estudiat els diferents tipus de màquina virtual, i com influeix la seva tipologia en la resta de processos del seu cicle de vida.

Les imatges de màquines virtuals, just a l'inici d'aquest cicle de vida, és a dir, durant l'arrancada o prèviament a aquesta, han de ser degudament preparades per tal que pugui proporcionar aquest espai d'execució virtual de manera adient, mitjançant un procés anomenat implantació. Per a aquest projecte s'han estudiat i implementat dos mecanismes d'implantació amb l'objectiu de cobrir el màxim nombre d'imatges possible, ja que els àmbits d'aplicació són diferents. El primer mecanisme estudiat duu a terme la implantació mitjançant la injecció de codi als paràmetres d'arrancada del nucli, pel que requereix que la imatge arranqui des d'un nucli extern. Per altra banda, el segon mecanismes es basa en la detecció i accés directe als sistemes de fitxers de la pròpia imatge. Les proves efectuades han permès comprovar la viabilitat d'ambdós mecanismes, així com mesurar el seu temps mig d'execució.

Una vegada preparades les imatges de màquina virtuals, són instanciades damunt els nodes físics per generar els diferents espais d'execució virtuals, segons les necessitats puntuals en cada moment dels treballs que es volen executar. Tots aquests treballs estan gestionats per

un component de l'arquitectura anomenat gestor d'execució. Una vegada els treballs han acabat d'executar-se, els espais d'execució virtuals poden ser aturats per alliberar recursos, o reciclats per a usar-se per altres treballs. Per a aquest projecte s'ha desenvolupat i provat un mecanisme per tal de determinar el millor moment per dur a terme l'aturada d'aquestes màquines virtuals. Aquest mecanisme, anomenat mecanisme del sentinella, permet aturar les màquines una vegada ha transcorregut un cert període d'inactivitat, que serveix com a marge per a permetre el reciclatge de la instància, en el cas que entrin nous treballs compatibles.

Tant la finalització d'un treball com l'aturada d'una màquina virtual marquen el final del cicle de vida d'una màquina. Tot i així, en aquest projecte també s'ha fet recerca sobre alguns estats intermedis opcionals que poden utilitzar-se per a dotar de més flexibilitat al processament de les tasques del grid. En particular, s'han estudiat les possibilitats de la suspensió, la hibernació i la migració a l'hora d'alliberar recursos de manera anticipada, per exemple per donar servei a tasques prioritàries o de computació urgent. Aquests estats intermedis s'integren en el cicle de vida de les màquines virtuals i són gestionats per la mateixa arquitectura.

Per altra banda, les línies obertes que es defineixen en aquest capítol es corresponen a aspectes tècnics del desenvolupament realitzat que han quedat pendents, en la major part dels casos degut a l'àmbit limitat de la recerca, i també a les funcionalitats de la resta de components de l'arquitectura que encara manquen per implementar:

- **Parametrització de l'entorn:** Un dels aspectes que ha quedat pendent en la tasca d'implantar els gestors de cues a les imatges de les màquines virtuals és la possibilitat de modificar-ne la seva configuració de manera dinàmica, sense haver de tornar a aplicar el procés d'implantació. Aquest objectiu es podria aconseguir modificant els guions de seqüència d'arrancada dels diferents serveis (de l'espai d'execució virtual o del sentinella, entre d'altres) per a que obtinguessin part de la seva configuració de l'exterior.

  Els dos mecanismes principals per accedir a informació exterior des d'una màquina virtual són dos: o bé mitjançant els paràmetres d'arrancada del nucli, o bé accedint a un recurs remot a través de la xarxa. En ambdós casos, el sistema pot adaptar el seu comportament segons la informació obtinguda. Tot i així, ambdues alternatives presenten limitacions: el paràmetres d'arrancada del nucli només seran útils a aquelles

màquines virtuals que no arranquin des d'un nucli encastat, mentre que l'accés a un recurs remot requerirà d'una connexió a la xarxa.

- **Definició de les imatges i dels treballs:** Actualment, cada solució de virtualització utilitza un llenguatge o format diferent i propietari per especificar les característiques i requeriments de les màquines virtuals i dels treballs. Darrerament, s'estan dedicant esforços a unificar aquest aspecte i s'estan impulsant llenguatges d'especificació formals que aspiren a esdevenir estàndards [E32].

  És d'esperar que en un futur, les diferents tecnologies i solucions de virtualització adoptin aquest format unificat, el que permetria millorar molt la interoperabilitat d'aquesta arquitectura amb altres hipervisors i gestors de cues, així com simplificar-ne la implementació.

- **Planificador dels recursos:** Al capítol 3, a l'apartat específic de clouds, ja s'ha esmentat que una de les mancances actuals dels sistemes que els gestionen és que disposen d'una pobre política d'assignació o distribució dels recursos, en la majoria de casos usen *round-robin*. Per a aquest projecte s'han desenvolupat i estudiat alguns mecanismes que permeten dur a terme diverses operacions sobre les imatges de les màquines virtuals, així com sobre les mateixes instàncies. La implantació d'un entorn d'execució virtual, la instanciació en un node físic, la determinació del moment idoni per l'aturada, o l'estudi dels efectes de la suspensió, la hibernació i la migració sobre el rendiment del grid han centrat la recerca en aquest projecte. Emperò, n'ha quedat fora de l'abast acabar d'integrar tots aquests mecanismes dins el component de planificació de l'arquitectura, de manera que, partint de la informació sobre l'estat del sistema, pugui respondre de manera autònoma sobre canvis en l'entorn: com una ampliació en el nombre d'espais d'execució, el reciclatge d'una instància o la migració cap a un altre node físic.

- **Reserva de recursos:** Un altre dels aspectes que ha quedat pendent és la implementació d'una funcionalitat que permeti la reserva de recursos per a tasques que hagin d'executar-se en un futur proper. Aquesta reserva de recursos estaria destinada a aquelles tasques de computació urgent, és a dir, amb uns requeriments de temps de resposta ajustats. Aquestes tasques no es poden permetre el luxe d'haver de passar per un procés de negociació i assignació de recursos quan entren al gestor de cues, i la

funcionalitat de reserva de recursos els permetria dur a terme aquesta negociació de manera anticipada. Per altra banda, la reserva de recursos dotaria d'una base de coneixement més àmplia al planificador que, per exemple, disposaria d'informació addicional a l'hora de decidir l'aturada d'un espai d'execució virtual, segons la previsió d'utilització dels recursos.

- **Multiplexació dels nodes:** Els nodes físics que disposen de hipervisor i, per tant, que poden instanciar màquines virtuals, podrien contenir més d'un espai d'execució virtual. Existeixen diverses maneres de multiplexar els nodes. En primer lloc es poden instanciar diverses màquines virtuals a sobre d'un mateix node físic, basant-se en el nombre de nuclis de processament de que disposi. Per altra banda, també existeix la possibilitat de configurar les instàncies de les màquines virtuals per a que permetin l'execució simultània de més d'un treball, compartint els recursos. Cal notar que, en aquest darrer cas, es perden part dels avantatges de l'ús dels espais d'execució virtuals, com la isolació de l'entorn o l'accés exclusiu als recursos.

6. Enllaços externs

E1. Basilisk II

http://basilisk.cebix.net/

E2. QEMU

http://www.qemu.org/

E3. Vmware

http://www.vmware.com/

E4. Xen

http://www.xen.org/

E5. KVM

http://www.linux-kvm.org/

E6. Extensions de virtualització Intel-VT (Vanderpool)

http://www.intel.com/pressroom/archive/releases/20050120comp.htm

http://www.intel.com/technology/virtualization/

E7. Extensions de virtualització AMD-V (Pacífica)

http://www.amd.com/us-en/Weblets/0,,7832_8366_7595~98372,00.html

http://www.amd.com/us-en/0,,3715_15781_15968,00.html

E8. Paravirtualització

http://www.xen.org/about/paravirtualization.html

http://www.vmware.com/interfaces/paravirtualization.html

E9. OpenPBS

http://www.openpbs.org/

E10. Sun Grid Engine

http://gridengine.sunsource.net/

E11. Condor

http://www.cs.wisc.edu/condor/

E12. Amazon EC2

http://aws.amazon.com/ec2/

E13. Eucalyptus

http://eucalyptus.cs.ucsb.edu/

E14. Grid Café

http://gridcafe.web.cern.ch/gridcafe

E15. Globus

http://www.globus.org

E16. PBS Professional

http://www.pbsgridworks.com/

E17. Google App Engine

http://appengine.google.com/

E18. GNU GRUB: GRand Unified Bootloader

http://www.gnu.org/software/grub/

E19. IBM: Inside the Linux boot process

http://www.ibm.com/developerworks/library/l-linuxboot/index.html

E20. POSIX: Portable Operating System Interface X

http://standards.ieee.org/regauth/posix/

E21. BASH: Bourne Again SHell

http://www.gnu.org/software/bash

E22. Perl

http://www.perl.org/

E23. Python

http://www.python.org/

E24. A look at System V initialization

http://www.freeos.com/articles/3243/

E25. Linux fastboot mailing list: Patch to add support for longer kernel command lines

https://lists.linux-foundation.org/pipermail/fastboot/2007-March/013170.html

E26. Linux documentation: Devices

http://kernel.org/doc/Documentation/devices.txt

E27. Linux documentation: Proc filesystem

http://kernel.org/doc/Documentation/filesystems/proc.txt

E28. Condor documentation: Job suspension

http://www.cs.wisc.edu/condor/manual/v7.3/3_5Policy_Configuration.html

E29. Condor documentation: VM hibernation

http://www.cs.wisc.edu/condor/manual/v7.3/3_3Configuration.html

E30. Google Docs

http://docs.google.com

E31. Globus Nimbus

http://workspace.globus.org/

E32. DMTF Standards for Virtualization Management

http://www.dmtf.org/initiatives/vman_initiative/

7. Bibliografia

[1]: Wolinsky, David Isaac and Agrawal, Abhishek and Boykin, P. Oscar and Davis, Justin R. and Ganguly, Arijit and Paramygin, Vladimir and Sheng, Y. Peter and Figueiredo, Renato J., «On the Design of Virtual Machine Sandboxes for Distributed Computing in Wide-area Overlays of Virtual Workstations» in *VTDC '06: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*. IEEE Computer Society: Washington, DC, USA, 2006, pp. 8.

[2]: Sotomayor, Borja and Keahey, Kate and Foster, Ian, «Combining batch execution and leasing using virtual machines» in *HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing*. ACM: Boston, MA, USA, 2008, pp. 87-96.

[3]: Krsul, Ivan and Ganguly, Arijit and Zhang, Jian and Fortes, Jose A. B. and Figueiredo, Renato J., «VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing» in *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*. IEEE Computer Society: Washington, DC, USA, 2004, pp. 7.

[4]: Andrés Cencerrado Barraqué, «Gestión de Máquinas Virtuales en entornos Grid - DDDAS», Universitat Autònoma de Barcelona, 2008.

[5]: R. J. Creasy, «The Origin of the VM 370 Time-sharing System» in *IBM Journal of Research & Development*, vol. 25, no. 5, pp. 483-490, September, 1981.

[6]: Adams, Keith and Agesen, Ole, «A comparison of software and hardware techniques for x86 virtualization» in *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*. ACM: San Jose, California, USA, 2006, pp. 2-13.

[7]: Robin, John Scott and Irvine, Cynthia E., «Analysis of the Intel Pentium's ability to support a secure virtual machine monitor» in *SSYM'00: Proceedings of the 9th conference on USENIX Security Symposium*. USENIX Association: Denver, Colorado, 2000, pp. 10.

[8]: John Fisher-Ogden, «Hardware support for efficient virtualization», 2006. *http://www.cse.ucsd.edu/jfisherogden/hardwareVirt.pdf*.

[9]: Jun Nakajima and Asit K. Mallick, «Hybrid-Virtualization—Enhanced Virtualization for Linux», 2007. *http://ols.108.redhat.com/2007/Reprints/nakajima-Reprint.pdf*.

[10]: Barham, Paul and Dragovic, Boris and Fraser, Keir and Hand, Steven and Harris, Tim and Ho, Alex and Neugebauer, Rolf and Pratt, Ian and Warfield, Andrew, «Xen and the Art of Virtualization» in *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. ACM: Bolton Landing, NY, USA, 2003, pp. 164-177.

[11]: Foster I, Kesselman C, «The Grid. Blueprint for a new computing environment». Morgan Kaufman, 1998.

[12]: Ian Foster, «What is the Grid? A Three Point Checklist», Argonne National Laboratory & University of Chicago, July 2002.

[13]: Rajkumar Buyya and Srikumar Venugopal, «A Gentle Introduction to Grid Computing and Technologies», Computer Society of India, July 2005.

[14]: Heinz Stockinger, «Defining the grid: a snapshot on the current view», in *The Journal of Supercomputing*, 2007, pp. 3-17.

[15]: Parvin Asadzadeh, Rajkumar Buyya1, Chun Ling Kei, Deepa Nayar, and Srikumar Venugopal, Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies, 2005

[16]: Vaquero, Luis M. and Rodero-Merino, Luis and Caceres, Juan and Lindner, Maik, «A break in the clouds: towards a cloud definition», in *SIGCOMM Comput. Commun. Rev.*, 2009, pp. 50-55.

[17]: Adabala, Sumalatha and Chadha, Vineet and Chawla, Puneet and Figueiredo, Renato and Fortes, Jose and Krsul, Ivan and Matsunaga, Andrea and Tsugawa, Mauricio and Zhang, Jian and Zhao, Ming and Zhu, Liping and Zhu, Xiaomin, «From Virtualized Resources to Virtual Computing Grids: The In-VIGO System», in *Future Generation Computer Systems*,

2005, pp. 896-909.

[18]: Santhanam, Sriya and Elango, Pradheep and Arpaci-Dusseau, Andrea and Livny, Miron, «Deploying virtual machines as sandboxes for the grid», in *WORLDS'05: Proceedings of the 2nd conference on Real, Large Distributed Systems*, 2005, pp. 7-12.

[19]: Kettimuthu, Rajkumar and Subramani, Vijay and Srinivasan, Srividya and Gopalasamy, Thiagaraja, «Selective Preemption Strategies for Parallel Job Scheduling», in *ICPP '02: Proceedings of the 2002 International Conference on Parallel Processing*, 2002, pp. 602.

[20]: Clark, Christopher and Fraser, Keir and Hand, Steven and Hansen, Jacob Gorm and Jul, Eric and Limpach, Christian and Pratt, Ian and Warfield, Andrew, «Live Migration of Virtual Machines», in *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, 2005, pp. 273-286.

[21]: Bradford, Robert and Kotsovinos, Evangelos and Feldmann, Anja and Schiöberg, Harald, «Live wide-area migration of virtual machines including local persistent state», in *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, 2007, pp. 169-179.

[22]: Santhanam, Sriya and Elango, Pradheep and Arpaci-Dusseau, Andrea and Livny, Miron, «Deploying virtual machines as sandboxes for the grid», in *WORLDS'05: Proceedings of the 2nd conference on Real, Large Distributed Systems*, 2005, pp. 7-12.