Introducció a la computació distribuïda

Ivan Rodero Castro
Francesc Guim Bernat

PID_00215392

UOC
Universitat Oberta de Catalunya

Índex

Introducció

En aquest mòdul didàctic estudiarem els conceptes més bàsics de sistemes distribuïts i dels dos tipus de sistemes distribuïts que tenen més relació i impacte en la computació d'altes prestacions. Aprendrem les característiques d'aquests sistemes però sobretot en veurem les diferències, limitacions i oportunitats com a plataformes per a altes prestacions.

En el primer apartat ens centrarem en els conceptes bàsics dels sistemes distribuïts i en les tecnologies bàsiques utilitzades per a implementar-los. En concret, estudiarem l'RPC, que és un mecanisme de crida a procediments remots; l'RMI, que permet la invocació de mètodes remots de manera transparent a partir del model d'objectes distribuïts, i finalment els serveis web, que és un mecanisme molt més obert i evolucionat i, entre altres coses, té un paper important en el desenvolupament de la computació en graella.

Un cop estudiades les tecnologies bàsiques per a sistemes distribuïts, ens centrarem en la computació en graella (*grid computing*), la qual es desenvolupa a partir de tecnologies de sistemes distribuïts com les que hem vist en el primer mòdul. En concret, estudiarem les característiques dels sistemes de computació en graella i després utilitzarem un exemple de programari intermediari (*middleware*) i un de metaplanificació (*meta-scheduling*) per a il·lustrar les funcionalitats més significatives d'aquests entorns.

Finalment estudiarem la computació en núvol (*cloud computing*). De la mateixa manera que veurem que la motivació de la computació en graella és la col·laboració entre institucions per a desenvolupar avenços científics a escala internacional, també veurem que el desenvolupament de la computació en núvol està basat en criteris econòmics i utilitza conceptes de mercat. Estudiarem tant les característiques principals dels sistemes en núvol, com ara l'elasticitat i el model de pagament de recursos per ús, com els diferents tipus de núvols, els diferents models de servei i el conceptes fonamentals de virtualització, utilitzant Xen com a cas d'estudi. Conclourem analitzant els possibles usos i reptes oberts per a utilitzar la computació en núvol per a computació d'altes prestacions.

Objectius

Els principals objectius que assolireu amb l'estudi d'aquest mòdul són el següents:

1. Conèixer els fonaments de la computació distribuïda i saber diferenciar-la de la computació d'altes prestacions tradicional.

2. Aprendre les tecnologies bàsiques de sistemes distribuïts com ara RPC, RMI i serveis web, que possibiliten el desenvolupament de sistemes distribuïts a escala com són la computació en graella o en núvol.

3. Entendre la motivació del desenvolupament de la computació en graella i saber identificar els components essencials, com són els conceptes d'*organització virtual*, de *programari* i de *metaplanificació*.

4. Distingir les diferències primordials entre els sistemes de computació en graella i els sistemes d'altes prestacions tradicionals.

5. Entendre la motivació del desenvolupament de la computació en núvol i saber identificar els components essencials, com són els conceptes d'*elasticitat*, de *pagar per ús* i de *virtualització*.

6. Aprendre els tipus de núvols més importants i els possibles models de servei.

7. Entendre les diferències entre els sistemes de computació en núvol, en graella i els sistemes d'altes prestacions tradicionals, i quines són algunes de les possibilitats per a aprofitar o combinar la computació en núvol amb la d'altes prestacions.

1. Fonaments de la computació distribuïda

En aquest apartat estudiarem els fonaments més bàsics de la computació distribuïda i ens centrarem en les tecnologies més rellevants que són la base i possibiliten el desenvolupament de la computació en graella i en núvol, que, tot i no seguir el model clàssic de computació d'altes prestacions, són els sistemes distribuïts que tenen un rol significatiu en la computació d'altes prestacions. Us encoratgem a fer una cerca sobre altres sistemes distribuïts, com ara els sistemes d'igual a igual (*peer-to-peer*), o models de computació que també es poden aplicar a sistemes distribuïts per a computació d'altes prestacions, com ara l'*autonomic computing*.

1.1. Introducció a la computació distribuïda

Un **sistema distribuït** es defineix com un conjunt de computadors autònoms connectats per una xarxa, i amb el programari distribuït adequat perquè el sistema sigui vist pels usuaris com una única entitat capaç de proporcionar capacitat de computació.

El desenvolupament dels sistemes distribuïts va arribar de la mà de les xarxes locals d'alta velocitat a principis de 1970. Recentment, la disponibilitat d'ordinadors personals d'altes prestacions, estacions de treball i ordinadors servidors ha donat com a resultat un desplaçament més gran cap als sistemes distribuïts en detriment dels ordinadors centralitzats multiusuari. Aquesta tendència s'ha accelerat pel desenvolupament de programari per a sistemes distribuïts, dissenyat per a donar suport al desenvolupament d'aplicacions distribuïdes. Aquest programari permet als ordinadors coordinar les seves activitats i compartir els recursos del sistema, és a dir, el maquinari, el programari i les dades.

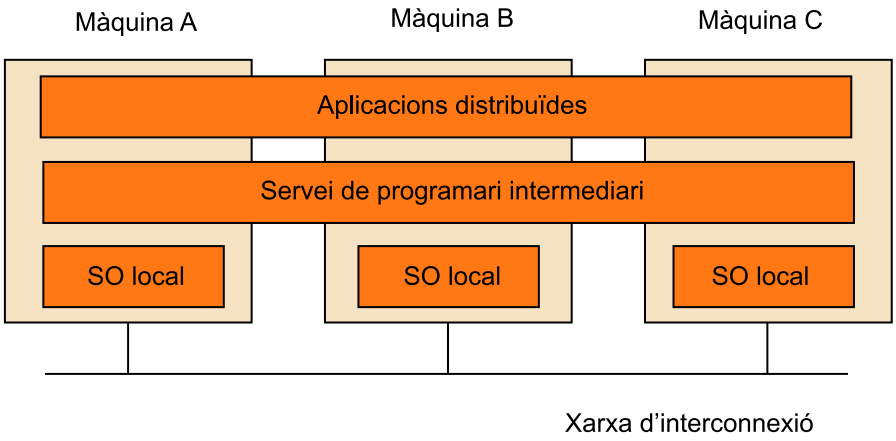
Els sistemes distribuïts s'implementen en diverses plataformes maquinari, des d'unes quantes estacions de treball connectades per una xarxa d'àrea local, fins a la Internet, que no deixa de ser un conjunt de xarxes d'àrea local i de gran abast interconnectades que enllacen milions d'ordinadors.

Les aplicacions dels sistemes distribuïts varien des de la provisió de capacitat de còmput a grups d'usuaris, fins a sistemes bancaris i comunicacions multimèdia, i contenen pràcticament totes les aplicacions comercials i tècniques dels ordinadors. Els requisits de les aplicacions inclouen un alt nivell de fiabilitat, seguretat contra interferències externes i privacitat de les dades. S'han de proveir accessos concurrents a bases de dades per part de molts usuaris, garan-

tir temps de resposta, proveir punts d'accés al servei que estan escampats geogràficament, etc., cosa que fa que els sistemes distribuïts tinguin un potencial molt gran per a les empreses per a créixer i fer negoci.

L'objectiu d'un sistema distribuït és integrar els recursos i serveis connectats per una xarxa d'interconnexió, tal com es pot veure en la figura 1. Des del punt de vista de l'usuari i de les aplicacions, un sistema distribuït proporciona una visió de màquina única i no es diferencia d'un de centralitzat. En canvi, des del punt de vista del dissenyador (el sistema com a gestor dels recursos) l'estructura interna està condicionada per la distribució física dels recursos.

Figura 1. Estructura d'un sistema distribuït genèric



El més habitual és que el sistema operatiu integri els serveis de xarxa que ofereixen protocols oberts de comunicació, com, per exemple, TCP i UDP. Sobre aquests protocols es disposen els suports addicionals per a la comunicació distribuïda, com és el cas d'RPC, RMI o DSM[1], i els serveis específics que proporcionen les propietats del sistema distribuït (serveis de programari intermediari o *middleware*), com és el cas de la gestió de temps, esdeveniments i estat global, sobre els quals es fonamenten les aplicacions.

[(1)]En anglès, *distributed shared memory*.

Activitat

Busqueu informació sobre DSM i penseu les similituds i les limitacions respecte d'altres models de memòria distribuïda.

Per a definir els sistemes distribuïts s'estableixen sis característiques principals:

- Compartició de recursos.
- Obertura.
- Concurrència.
- Escalabilitat.
- Tolerància a fallades.
- Transparència.

1.1.1. Compartició de recursos

El terme *recurs* és bastant abstracte, però és el que caracteritza més bé el ventall d'entitats que poden compartir en un sistema distribuït. El ventall s'estén des de components maquinari com discos i impressores, fins a elements programari com fitxers, bases de dades i altres objectes de dades.

La idea de compartició de recursos no és nova ni apareix en el marc dels sistemes distribuïts. Els sistemes multiusuaris clàssics des de sempre han proveït de compartició de recursos els seus usuaris. No obstant això, els recursos d'un ordinador multiusuari es comparteixen de manera natural entre tots els seus usuaris. Per contra, els usuaris d'estacions de treball monousuari o ordinadors personals dins d'un sistema distribuït no obtenen automàticament els beneficis de la compartició de recursos.

Els recursos en un sistema distribuït estan físicament encapsulats en un dels computadors i només s'hi pot accedir per altres computadors mitjançant comunicació via xarxa. Perquè la compartició de recursos sigui efectiva, ha de ser gestionada per un programa que ofereixi una interfície de comunicació que permeti accedir al recurs, manipular-lo i actualitzar-lo d'una manera fiable i consistent. Aquí sorgeix el concepte de *gestor de recursos*.

Un **gestor de recursos** és un mòdul programari que gestiona un conjunt de recursos d'un tipus en particular. Cada tipus de recurs requereix alguns mètodes i polítiques específics juntament amb requisits comuns per a tots plegats. Aquests requisits inclouen la provisió d'un esquema de noms per a cada classe de recurs, la permissió d'accedir als recursos individuals des de qualsevol localització, la traducció del nom del recurs a adreces de comunicació i la coordinació dels accessos concurrents que canvien l'estat dels recursos compartits per a mantenir la consistència.

Un sistema distribuït es pot veure de manera abstracta com un conjunt de gestors de recursos i un conjunt de programes que fan servir els recursos. Els usuaris dels recursos es comuniquen amb els gestors dels recursos per accedir als recursos compartits del sistema. Aquesta perspectiva ens porta a dos models de sistemes distribuïts: el model client-servidor i el model basat en objectes.

1.1.2. Obertura

Un sistema informàtic és **obert** si el sistema pot ser estès de diverses maneres. Un sistema pot ser obert o tancat pel que fa a extensions maquinari (afegir perifèrics, memòria, interfícies de comunicació, etc.) o respecte a les extensions programari (afegir característiques al sistema operatiu, protocols de comunicació i serveis de compartició de recursos, etc.).

L'obertura dels sistemes distribuïts es determina primàriament pel grau en què es poden afegir nous serveis de compartició de recursos sense perjudicar ni duplicar els existents. Vegem algunes de les característiques dels sistemes distribuïts que cal tenir en compte respecte a l'obertura:

- Les interfícies programari clau del sistema estan clarament especificades i es posen a disposició dels desenvolupadors, és a dir, les interfícies es fan públiques.

- Els sistemes distribuïts oberts es basen en la provisió d'un mecanisme uniforme de comunicació entre processos i interfícies publicades per a accedir a recursos compartits.

- Els sistemes distribuïts oberts es poden construir a partir de maquinari i programari heterogeni, possiblement provinent de venedors diferents. Però la conformitat de cada component amb l'estàndard publicat ha de ser acuradament comprovada i certificada si es vol evitar tenir problemes d'integració.

1.1.3. Concurrència

Quan hi ha diversos processos en una única màquina diem que s'estan **executant concurrentment**. Si el computador està equipat amb un únic processador central, la concurrència té lloc entrellaçant l'execució dels diferents processos. Si el computador té N processadors, es poden estar executant estrictament alhora fins a N processos.

En els sistemes distribuïts hi ha moltes màquines, cadascuna amb un o més processadors centrals. És a dir, si hi ha M ordinadors en un sistema distribuït amb un processador central cadascun, es poden estar executant en paral·lel fins a M processos.

En un sistema distribuït que està basat en el model de compartició de recursos, la possibilitat d'execució paral·lela passa per dues raons:

1) Hi ha molts usuaris que interactuen simultàniament amb programes d'aplicació.

2) Hi ha molts processos servidor que s'executen concurrentment, cadascun responent a diferents peticions de processos client.

El primer cas és menys conflictiu, ja que normalment les aplicacions d'interacció s'executen aïlladament a l'estació de treball de l'usuari i no entren en conflicte amb les aplicacions executades a les estacions de treball d'altres usuaris.

El segon cas sorgeix a causa de l'existència d'un o més processos servidor per a cada tipus de recurs. Aquests processos s'executen en diferents màquines, de manera que s'executen en paral·lel diversos servidors, juntament amb diversos programes d'aplicació. Les peticions per a accedir als recursos d'un servidor especificat poden ser encuades al servidor i processades seqüencialment o bé poden ser processades concurrentment per múltiples instàncies del procés de gestor de recursos. Quan passa això, els processos servidor han de sincronitzar les accions per a assegurar-se que no hi ha conflictes. La sincronització ha de ser planejada acuradament per a assegurar que no es perden els beneficis de la concurrència.

1.1.4. Escalabilitat

Els sistemes distribuïts operen de manera efectiva i eficient en moltes escales diferents. L'escala més petita consisteix, per exemple, en dues estacions de treball i un servidor de fitxers, mentre que un sistema distribuït construït al voltant d'una xarxa d'àrea local simple podria contenir diversos centenars d'estacions de treball, diversos servidors de fitxers, servidors d'impressió i altres servidors de propòsit específic. Sovint es connecten diverses xarxes d'àrea local per a formar-ne de més grans, les quals podrien contenir molts milers d'ordinadors que formen un únic sistema distribuït, i permetre així que els recursos siguin compartits entre tots plegats.

Tant el programari de sistema com el d'aplicació no haurien de canviar quan l'escala del sistema augmenta. La necessitat d'escalabilitat no és solament un problema de prestacions de xarxa o de maquinari, sinó que està íntimament lligada a tots els aspectes del disseny dels sistemes distribuïts. El disseny del sistema s'ha de fer partint de la idea que caldrà que escali o en cas contrari hi haurà moltes limitacions.

La demanda d'escalabilitat en els sistemes distribuïts ha conduït a una filosofia de disseny en què qualsevol recurs, sia maquinari o programari, es pot estendre per a proporcionar servei a tants usuaris com es vulgui. Això vol dir

que, si la demanda d'un recurs creix, hauria de ser possible estendre el sistema per a donar servei. Per exemple, la freqüència amb la qual s'accedeix als fitxers creix quan s'incrementa el nombre d'usuaris i estacions de treball en un sistema distribuït. Llavors, ha de ser possible afegir servidors per a evitar el coll d'ampolla que es produiria si un sol servidor de fitxers hagués de gestionar totes les peticions d'accés als fitxers. En aquest cas, el sistema ha d'estar dissenyat de manera que permeti treballar amb fitxers replicats en diferents servidors, amb les consideracions de consistències que això comporta.

Quan la mida i complexitat de les xarxes creix, un objectiu primordial és dissenyar programari distribuït, que continuarà essent eficient i útil amb aquestes noves configuracions de la xarxa.

Resumint, el treball necessari per a processar una petició per a accedir a un recurs compartit hauria de ser pràcticament independent de la mida de la xarxa. Les tècniques necessàries per a aconseguir aquests objectius inclouen l'ús de dades replicades, de tècniques relacionades amb memòries cau i de múltiples servidors per a gestionar les tasques per tal de permetre aprofitar la concurrència i per tant obtenir més productivitat.

1.1.5. Tolerància a fallades

Els sistemes informàtics de vegades fallen. Quan es produeixen errors en el programari o el maquinari, els programes poden produir resultats incorrectes o es poden aturar abans d'acabar la computació que estaven fent.

El disseny de **sistemes tolerants a fallades** es basa en dues qüestions, complementàries entre si: redundància del maquinari (ús de components redundants) i recuperació del programari (disseny de programes que siguin capaços de recuperar-se dels errors).

En els sistemes distribuïts, la redundància es pot plantejar en un gra més fi que el maquinari, es poden replicar els servidors individuals que són essencials per a l'operació continuada d'aplicacions crítiques. La recuperació del programari té relació amb el disseny de programari que sigui capaç de recuperar l'estat de les dades permanents abans que es produís la fallada.

La disponibilitat en els sistemes distribuïts també té una gran importància i moltes vegades la manca d'aquesta disponibilitat està relacionada amb fallades de maquinari. La disponibilitat d'un sistema és una mesura de la proporció de temps que està disponible per a usar-lo. Una simple fallada en una màquina multiusuari té com a resultat la no-disponibilitat del sistema per a tots els usuaris. Quan un dels components d'un sistema distribuïts falla, només es veu

afectat el treball que estava fent el component avariat. Un usuari es podria desplaçar a una altra estació de treball i un procés servidor es podria executar en un altre computador.

1.1.6. Transparència

La **transparència** es defineix com l'ocultació a l'usuari i al programador d'aplicacions de la separació dels components d'un sistema distribuït, de manera que el sistema es percep com un tot, en comptes d'un conjunt de components independents.

La transparència exerceix una gran influència en el disseny del programari de sistema. Podem parlar de vuit formes de transparència:

1) Transparència d'accés: permet l'accés als objectes d'informació remots de la mateixa manera que als objectes d'informació locals.

2) Transparència de localització: permet l'accés als objectes d'informació sense conèixer-ne la localització.

3) Transparència de concurrència: permet que diversos processos operin concurrentment utilitzant objectes d'informació compartits i de manera que no hi hagi interferència entre uns i altres.

4) Transparència de replicació: permet utilitzar múltiples instàncies dels objectes d'informació per a incrementar la fiabilitat i les prestacions sense que els usuaris o els programes d'aplicació hagin de conèixer l'existència de les rèpliques.

5) Transparència de fallades: permet als usuaris i programes d'aplicació completar les tasques tot i l'ocurrència de fallades en el maquinari o el programari.

6) Transparència de migració: permet el moviment d'objectes d'informació dins d'un sistema sense afectar els usuaris o els programes d'aplicació.

7) Transparència de prestacions: permet que el sistema sigui reconfigurat per a millorar les prestacions mentre la càrrega varia.

8) Transparència d'escalat: permet l'expansió del sistema i de les aplicacions sense canviar l'estructura del sistema o els algoritmes de l'aplicació.

Les dues més importants són les transparències d'accés i de localització; la presència o absència d'aquestes formes afecta fortament la utilització dels recursos distribuïts. Sovint són anomenades totes dues **transparències de xarxa**.

1.2. Model client-servidor

El model client-servidor d'un sistema distribuït és el model més conegut i més àmpliament adoptat en l'actualitat. Hi ha un conjunt de processos del servidor, en què cadascun actua com un gestor de recursos per a un conjunt de recursos d'un tipus, i un conjunt de processos client, en què cadascun porta a terme una tasca que requereix accés a alguns recursos maquinari i programari compartits.

Els gestors de recursos podrien necessitar accedir a recursos compartits gestionats per altres processos, de manera que hi ha alguns processos que poden ser tant clients com servidors. En el model client-servidor, tots els recursos compartits són mantinguts i gestionats pels processos servidor. Els processos client fan peticions als servidors quan necessiten accedir a algun recurs. Si la petició és vàlida, el servidor porta a terme l'acció requerida i envia una resposta al procés client.

El model client-servidor ens dóna un enfocament efectiu i de propòsit general per a la compartició d'informació i de recursos en els sistemes distribuïts. El model pot ser implementat en una gran varietat d'entorns programari i maquinari. Els computadors que executen els programes client i servidor poden ser de molts tipus i no hi ha la necessitat de distingir-los ja que els processos client i servidor fins i tot poden ser a la mateixa màquina.

En aquesta aproximació del model client-servidor, cada procés servidor podria ser vist com un proveïdor centralitzat dels recursos que gestiona. La provisió de recursos centralitzada no és desitjable en els sistemes distribuïts, però. Per això es fa una distinció entre els serveis proporcionats als clients i els servidors encarregats de proveir aquests serveis. Un servei es considera una entitat abstracta que pot ser proveïda per diversos processos servidor que funcionen en uns quants computadors i cooperen mitjançant la xarxa.

El model client-servidor s'ha estès i utilitzat en els sistemes actuals amb serveis que gestionen molts tipus diferents de recursos compartits com ara correu electrònic i missatges de notícies, fitxers, sincronització de rellotges, emmagatzematge en disc, impressores i fins i tot les interfícies gràfiques d'usuari. Però no és possible que tots els recursos que hi ha en un sistema distribuït siguin gestionats i compartits d'aquesta manera; hi ha alguns tipus de recursos que han de romandre locals a cada computador amb vista a una més bona eficiència, com ara la memòria RAM, el processador o la interfície de xarxa local. Aquests recursos clau són gestionats separadament per un sistema operatiu a cada computador; només poden ser compartits entre processos localitzats en el mateix computador.

1.2.1. Programari intermediari

El programari distribuït requerit per a facilitar les interaccions client-servidor s'anomena **programari intermediari** (*middleware*). L'accés transparent a serveis i recursos distribuïts mitjançant una xarxa d'interconnexió es fa amb el programari intermediari, que serveix com a marc per a les comunicacions entre els components client i servidor d'un sistema.

El programari intermediari defineix la interfície que usen els clients per a demanar un servei a un servidor, la transmissió física de la petició via xarxa i la devolució de resultats des del servidor al client. Alguns dels exemples de programari intermediari estàndard per a dominis específics són: ODBC per a bases de dades, HTTP i SSL per a Internet, i CORBA i RMI per a objectes distribuïts.

El programari intermediari fonamental o genèric és la base dels sistemes client-servidor. Els serveis d'autenticació en xarxa, crides a procediments remots, sistemes de fitxers distribuïts i serveis de temps en xarxa es consideren una part del programari intermediari genèric. El programari intermediari específic per a un domini complementa el programari intermediari genèric amb vista a aplicacions molt més específiques. El protocol de comunicacions més utilitzat pel programari intermediari, tant genèric com específic, és TCP/IP per la gran difusió que té.

1.3. Crida a procediment remot (RPC)

El pas de missatges permet expressar el model client-servidor explicitant un protocol de petició-resposta sobre el servei al qual es vol accedir. En canvi, l'accés a recursos en els sistemes operatius tradicionals mitjançant la interfície de crides al sistema es basa en una semàntica de crida-retorn a funcions. Essent la transparència en la ubicació dels recursos un objectiu fonamental en els sistemes distribuïts, resulta evident que el pas de missatges no és una base semàntica adequada per a l'accés als recursos remots.

El mecanisme de crida a procediment remot (RPC[2]) mira d'eliminar aquest salt semàntic. Un procés client especifica l'accés a un servei mitjançant una sintaxi de crida a funció, com, per exemple:

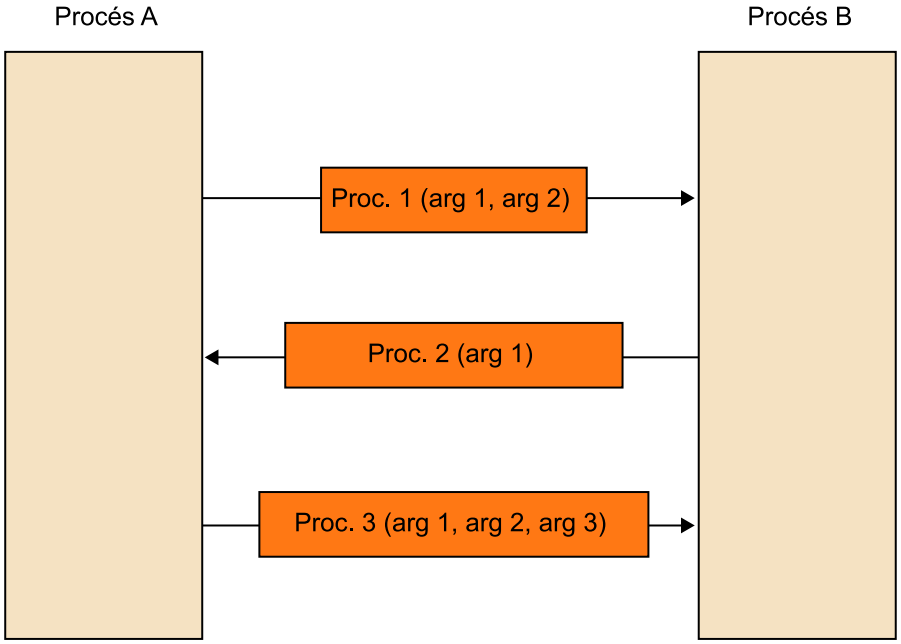
(2)De l'anglès *remote procedure call*.

```
resultat = servei_rpc (paràmetres)
```

Un servei proporcionat per un servidor no és més que un conjunt d'operacions disponibles per als clients. L'accés al servei es fa mitjançant un protocol de peticions i respostes amb crides bloquejants. Un exemple d'això és el sistema

de fitxers en què el servidor manté com a recurs compartit els fitxers i sobre el qual es poden fer diverses operacions, com, per exemple, crear, obrir o llegir. La figura 2 mostra un exemple genèric de crides RPC.

Figura 2. Exemples de crides a procediments remots



Els mecanismes **RPC** s'encarreguen de proporcionar als clients una capa d'abstracció per a cridar procediments remots (operacions) i així obtenir serveis.

D'aquesta manera, el procediment especificat s'executa en un altre procés d'una altra màquina (servidor). L'objectiu d'RPC és mantenir la semàntica de la crida a procediments normals en un entorn d'implementació totalment diferent. L'avantatge està en el fet que el desenvolupador només s'ha de preocupar de les interfícies amb què funciona el servidor. Per a especificar aquestes interfícies es disposa d'un llenguatge de definició d'interfícies (IDL[3]).

[(3)]De l'anglès *interface definition language*.

Els sistemes RPC disposen de mecanismes d'RPC integrats en un llenguatge de programació particular que inclou a més una notació per a definir interfícies entre clients i servidors (IDL específic). Un IDL permet definir el nom de les operacions amb què funciona el servidor i els paràmetres d'aquestes operacions (tipus i adreça). També s'han de proveir mecanismes per a maneig d'excepcions, garantir l'execució de les operacions, i també la detecció de fallades. Tot això de la manera més transparent possible.

El programari intermediari amb què funciona RPC té tres tasques fonamentals:

1) Configurar el processament relacionat amb les interfícies: integrar RPC en l'entorn de programació, empaquetament[4] / desempaquetament[5] i atendre les peticions en el procediment adequat.

(4)En anglès, *marshalling*.

(5)En anglès, *unmarshalling*.

2) Gestionar les comunicacions.

3) Enllaçar[6]: localitzar el servidor d'un servei.

(6)En anglès, *binding*.

La interfície no és més que un nom de procediment acordat entre client i servidor. Aquest nom viatja dins del missatge RPC transmès per la xarxa. A la part del client hi ha un procediment d'estub (*stub*) encarregat d'empaquetar i desempaquetar els arguments de la crida i convertir la crida local en una de remota. Això implica enviar un missatge, esperar la resposta i retornar els resultats. A la part del servidor hi ha el despatxador, juntament amb el conjunt de procediments d'estub de servidor, que tenen una missió similar als de la part del client. El despatxador selecciona el procediment d'estub adequat a partir del nom de procediment requerit.

El compilador d'IDL genera els procediments d'estub de client i de servidor, les operacions d'empaquetament i desempaquetament i els fitxers de capçalera necessaris. Les peticions dels clients es fan amb un nom de servei. En darrera instància han de ser dirigides a un port al servidor. En un sistema distribuït, un *binder* és un servei que manté una taula que conté correspondències de noms de serveis amb ports de servidor. Es tracta de fet d'un servei de noms. Importar un servei és demanar al *binder* que busqui el nom de la interfície i retorni el port del servidor. Exportar un servei és registrar pensant en el *binder*. El *binder* ha de ser en un port ben conegut o, si més no, s'hi ha de poder localitzar.

Una implementació bastant habitual d'RPC és la de SUN, que incorpora tot un conjunt de primitives per a treballar amb RPC en llenguatge C. Disposa d'una representació neutral de les dades (XDR[7]) per a fer-ne l'empaquetament.

(7)De l'anglès *external data representation*.

El compilador d'IDL, que s'anomena *rpcgen*, genera els procediments d'estub, el procediment *main* del servidor i el despatxador, el codi de conversió dels paràmetres a XDR, i també els fitxers de capçalera corresponents.

El procediment complet per a l'execució d'una RPC és el següent:

1) A partir del nom del servei remot el client localitza el servidor amb què funciona el servei.

2) S'empaqueten els arguments en un format estàndard (per exemple, XDR20) per a formar el cos d'un missatge, és a dir, se serialitzen[8].

(8)En anglès, *marshaling*.

3) S'utilitza la interfície de xarxa del sistema operatiu per a enviar un missatge al servidor que conté la petició del servei (per exemple, mitjançant UDP/IP).

4) El procés client queda bloquejat en la recepció de la resposta.

5) En el node destí, en rebre el missatge, el sistema operatiu desbloqueja el servidor. Si aquest servidor és multifil, un fil d'execució es farà càrrec de la petició.

6) S'executa el mecanisme que desempaqueta el missatge (desserialització) per a obtenir els paràmetres de la petició de servei i la identificació de l'origen.

7) S'executa la funció associada al servei sol·licitat, que ha estat instal·lada pel servidor en la inicialització.

8) La funció del servidor s'executa mitjançant crides al sistema local.

9) Quan acaba la funció associada al servei se serialitza el resultat.

10) S'envia el missatge amb el resultat al client.

11) El sistema operatiu del node del client desbloqueja aquest client en la recepció del resultat.

12) El client executa la desserialització del resultat.

13) Es retorna el valor al programa.

1.4. Invocació de mètodes remots (RMI)

La invocació de mètodes remots (RMI[9]), correspon al model de programació orientada a objectes, al contrari de les RPC, que pertanyen al model de programació procedimental. Primer veurem els conceptes relacionats amb la invocació de mètodes del model d'objectes i després ens centrarem en el model d'objectes distribuïts.

(9)De l'anglès *remote method invocation*.

1.4.1. Model d'objectes

En els llenguatges de programació orientada a objectes, com C++ i Java, un objecte, que encapsula un conjunt de dades i de mètodes, es comunica amb altres objectes invocant els mètodes d'aquests objectes, que en general accepten arguments i retornen resultats.

Tot i que aquests llenguatges proporcionen formes per a permetre referenciar directament les variables dels objectes, en el model d'objectes distribuïts, que veurem després, les variables només poden accedir per mitjà de mètodes.

Als objectes s'hi accedeix per referència. En la invocació d'un mètode d'un objecte (que anomenem *objecte receptor*), cal proporcionar la referència al receptor, el mètode i els arguments de la invocació. Les referències a objectes es poden assignar a variables, passar com a arguments o retornar com a resultats d'una invocació. Una interfície defineix el format d'accés a un conjunt de mètodes, és a dir, els seus noms, els tipus dels arguments, els valors de retorn i les excepcions (però no la implementació dels mètodes).

En la invocació d'un mètode, l'objecte receptor executa el mètode i retorna el control, i eventualment torna un resultat. Com a conseqüència d'això, l'estat del receptor pot canviar. També es poden desencadenar invocacions a altres objectes. Durant l'execució del mètode també es poden produir condicions d'error (excepcions).

1.4.2. Model d'objectes distribuïts

En un sistema distribuït, els objectes de les aplicacions poden estar repartits entre els nodes del sistema. Com passa en les RPC, la invocació de mètodes remots normalment es fa mitjançant el model client-servidor, en què els objectes receptors són gestionats pels servidors d'aquests objectes. La invocació manté la transparència en la ubicació de l'objecte.

En el **model d'objectes distribuïts**, els processos consten d'objectes que es comuniquen mitjançant invocacions locals o remotes. Les locals són invocacions a mètodes del mateix objecte, mentre que les remotes són invocacions a mètodes d'objectes d'altres processos, tant si són al mateix node com a d'altres del sistema.

Alguns objectes només poden rebre invocacions locals, mentre que d'altres, els objectes remots, poden rebre tant invocacions locals com remotes. Per a invocar un mètode d'un objecte remot, un objecte ha de tenir accés a la referència d'objecte remot del receptor, que és un identificador únic al sistema distribuït.

Cada objecte remot té una interfície remota que especifica quins dels seus mètodes es poden invocar remotament. Aquests mètodes estan implementats per la classe de l'objecte remot. En Java RMI, les interfícies remotes es defineixen com qualsevol altra interfície de Java, sense haver de fer res més que ampliar la interfície "Remote". CORBA proporciona un llenguatge de definició d'interfícies (CORBA IDL) que permet que les classes dels objectes remots i els programes client estiguin programats en llenguatges diferents. En una invocació remota es poden produir, a més de les generals, excepcions relacionades amb la naturalesa remota de la invocació, com, per exemple, *timeouts*.

1.4.3. Java RMI

Java RMI amplia el model d'objectes de Java per funcionar amb objectes distribuïts de manera integrada en el llenguatge, i fa la invocació de mètodes remots transparent excepte pel fet que el client hagi de tractar les excepcions remotes i el servidor de definir com a "Remote" la interfície de l'objecte remot.

Les interfícies remotes es defineixen ampliant la interfície "Remote", la qual és proporcionada pel paquet "java.rmi". Els paràmetres d'un mètode són d'entrada, i la sortida es proporciona en el resultat de la invocació. Els objectes remots es passen per referència i els locals per valor, mitjançant serialització. Quan el receptor no disposa de la implementació de l'objecte que se li passa per valor, la màquina virtual Java proporciona la baixada automàtica de la classe corresponent.

Els nodes que contenen objectes remots proporcionen un servei de noms que emmagatzema les referències d'objectes remots, el registre d'objectes remots ("rmiregistry").

En una aplicació distribuïda en Java RMI, cal definir les interfícies remotes i implementar els objectes remots i els clients. Un cop compilats els fitxers font, l'aplicació es munta de la manera següent:

1) Es creen els servidors intermediaris o *proxies* (*estubs*, en terminologia Java RMI) de les classes remotes mitjançant el compilador de RMI ("rmic").

2) S'especifica l'accés a les classes i s'estableix una política de seguretat.

3) Es posa en marxa el registre d'objectes remots com un procés del servidor ("rmiregistry").

4) Es posa en marxa el servei remot i es llancen els clients.

Implementació d'un mètode que suma dos números en Java RMI

A continuació mostrem un exemple d'implementació d'un mètode que suma dos números en Java RMI.

Primer es defineix la interfície que declara el mètode remot, el qual té dos números com a arguments i retorna la suma.

```
import java.rmi.*;

public interface AddServerIntf extends Remote {
   double add(double d1, double d2) throws RemoteException;
}
```

Després s'implementa el codi corresponent a la interfície remota i el servidor.

```
import java.rmi.*;
import java.rmi.server.*;

public class AddServerImpl extends UnicastRemoteObject
   implements AddServerIntf {
```

```
    public AddServerImpl() throws RemoteException {
    }
    public double add(double d1, double d2) throws RemoteException {
       return d1 + d2;
    }
}
```

El codi següent s'encarrega de fer que els objectes siguin disponibles per nodes remots, és a dir, que actualitza el registre RMI (mitjançant "rebind").

```
import java.net.*;
import java.rmi.*;

public class AddServer {
   public static void main(String args[]) {
      try {
         AddServerImpl addServerImpl = new AddServerImpl();
         Naming.rebind("AddServer", addServerImpl);
      }
      catch(Exception e) {
         System.out.println("Exception: " + e);
      }
   }
}
```

Finalment cal implementar un programa client que serà l'encarregat d'utilitzar la interfície remota i acabar cridant el mètode de l'objecte remot. Aquest client té tres arguments: l'adreça IP o el nom del servidor remot, i els dos números que s'han de sumar.

```
import java.rmi.*;
public class AddClient {
   public static void main(String args[]) {
      try {
         String addServerURL = "rmi://" + args[0] + "/AddServer";
         AddServerIntf addServerIntf =
            (AddServerIntf)Naming.lookup(addServerURL);
         System.out.println("The first number is: " + args[1]);
         double d1 = Double.valueOf(args[1]).doubleValue();
         System.out.println("The second number is: " + args[2]);

         double d2 = Double.valueOf(args[2]).doubleValue();
       System.out.println("The sum is: " + addServerIntf.add(d1, d2));
      }
      catch(Exception e) {
         System.out.println("Exception: " + e);
      }
   }
}
```

1.5. Serveis web

El concepte de servei web[10] ha evolucionat des de la definició inicial. Els inicis dels serveis web els podem trobar en els sistemes distribuïts, amb CORBA i DCOM, però cap dels dos sistemes no va acabar generalitzant-se a causa dels problemes d'aplicació en la Internet real. Més endavant el W3C predefiniria els serveis web amb el protocol SOAP[11], que forma part de la capa de missatgeria. Des que es va crear se n'ha estès ràpidament el funcionament i ha estat fonamental per a teixir tant el Web 2.0 com altres infraestructures com ara la computació en graella[12].

(10) En anglès, *web service*.

(11) De l'anglès *simple object access protocol*.

(12) En anglès, *grid computing*.

Actualment, els serveis SOAP destinats a aplicacions distribuïdes conviuen amb noves formes més simples i no estàndards pel més bon rendiment que tenen en relació amb la implementació de SOAP. XML-RPC, JSON-RPC, PHP-RPC són els nous mètodes/protocols/formats més destacats que trobem en els serveis web a les interfícies, per exemple, de Facebook, Google i Yahoo.

Entre les diverses organitzacions que s'encarreguen de vetllar pels estàndards i l'arquitectura web destaquen el WWWC (W3C[13]) i OASIS[14].

(13)De *World Wide Web Consortium*.

(14)De *Organization for the Advancement of Structured Information Standards*.

El W3C defineix els serveis web de la manera següent:

"Un servei web és un sistema de programari dissenyat per a donar suport a la interacció interoperable de màquina a màquina sobre una xarxa. Té una interfície descrita en un format processable per màquina, específicament WSDL. Hi ha altres sistemes que interactuen amb el servei web d'una manera prescrita per la seva descripció usant missatges SOAP, típicament transmès per HTTP amb una serialització XML en conjunció amb altres normes relacionades amb la web."

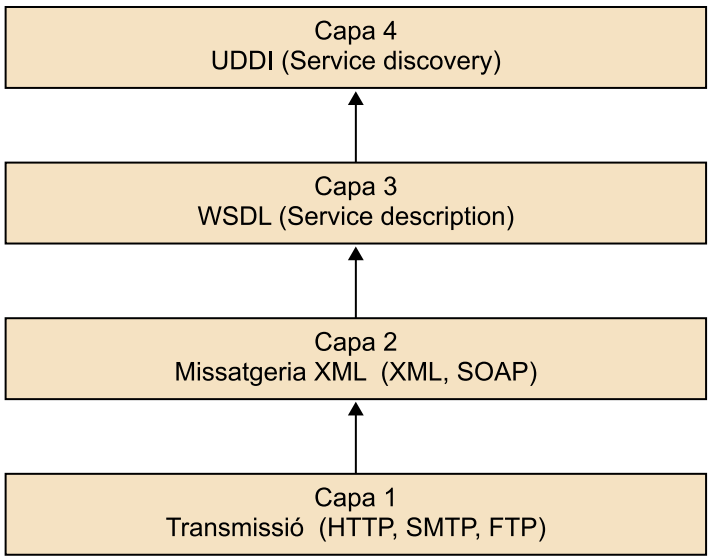
El W3C es pot definir més clarament com una manera estandarditzada d'integrar aplicacions web mitjançant l'ús d'XML, SOAP, WSDL i UDDI, els quals permeten la comunicació entre aplicacions o components d'aquestes aplicacions de manera estàndard mitjançant protocols comuns (típicament HTTP) i de manera totalment independent del llenguatge de programació, la plataforma d'implantació, el sistema operatiu o el format de presentació. L'èxit de la interoperabilitat s'assoleix amb l'adopció de protocols i estàndards oberts.

En un intent d'estandardització global, el W3C proposa la seva visió dels serveis web, especificant la *web services protocol stack* (figura 3), que és una pila de protocols per a serveis web. Tot i el pes i la importància del W3C com a organisme d'estandardització en els serveis web, entre altres tecnologies, és ben cert que hi ha molts fabricants que proposen petits canvis a aquesta especificació.

Web services protocol stack

És un conjunt de protocols i estàndards per a xarxes (Internet, intranet, etc.) utilitzats per a definir, localitzar, implementar i fer que un servei web interactuï amb altres.

Figura 3. *Web services protocol stack*



La capa de transport és responsable del transport dels missatges entre les aplicacions de xarxa i els protocols com ara HTTP, HTTPS, SMTP, FTP, JABBER, IIOP o BEEP. A la pràctica, la gran majoria de serveis web estan disponibles per HTTP o HTTPS.

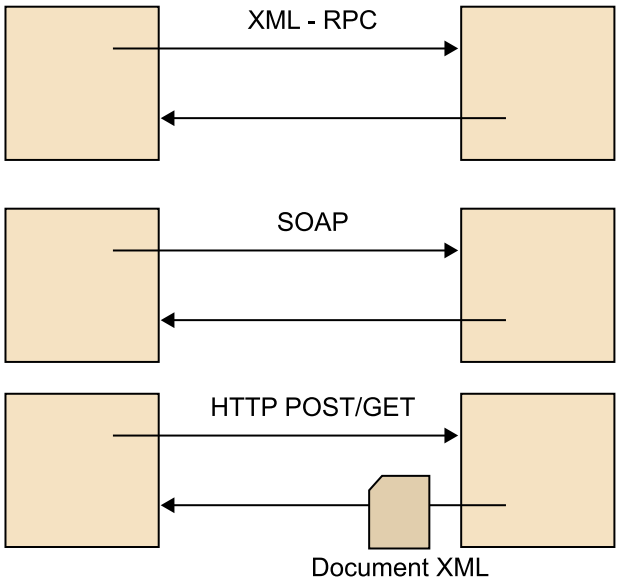
La **missatgeria XML** consisteix en un llenguatge de marques ampliable. És un estàndard per a descriure dades i crear etiquetes. Les característiques especials són la independència de dades i la separació dels continguts de la seva representació. És un metallenguatge que permet dissenyar un llenguatge propi d'etiquetes per a múltiples tipus de documents. Els documents XML estan formats per unitats d'emmagatzematge, anomenades *entitats*, que contenen dades analitzades[15] o sense analitzar[16]. Les dades analitzades estan formades per caràcters. Alguns d'aquests caràcters formen les dades del document i la resta formen les etiquetes.

[15] En anglès, *parsed*.

[16] En anglès, *unparsed*.

Les etiquetes codifiquen la descripció de l'estructura lògica i d'emmagatzematge del document. XML proporciona un mecanisme per a imposar restriccions a l'estructura lògica i d'emmagatzematge. XML es va convertir en l'estàndard de les comunicacions per Internet. El que abans es transmetia en formats propietaris ara és fàcilment interoperable gràcies a XML. En l'especificació del W3C només es preveu XML i SOAP com a capa de missatgeria (figura 4).

Figura 4. Missatgeria XML



El **WSDL**[17] és un llenguatge de descripció de serveis web. És una especificació XML per a la construcció del document de descripció del servei web. Quan diem que és una especificació XML volem dir que el document WSDL està escrit en XML.

(17)De l'anglès *web service description language*.

El fitxer WSDL identifica els mètodes, funcions i paràmetres necessaris per a invocar un determinat servei i descriu informació crítica que el client del servei web ha de saber, com:

- El nom del servei, incloent-hi l'URN[18].
- La localització del servidor, normalment una adreça URL[19] per HTTP.
- Els mètodes disponibles per a ser invocats.
- Els paràmetres d'entrada i sortida per a cadascun dels mètodes.

(18)De l'anglès *uniform resource name*.

(19)De l'anglès *uniform resource locator*.

Exemple de fitxer WSDL

A continuació mostrem un exemple de fitxer WSDL que descriu el servei del típic "Hello World" en el qual es defineix el mètode "sayHello".

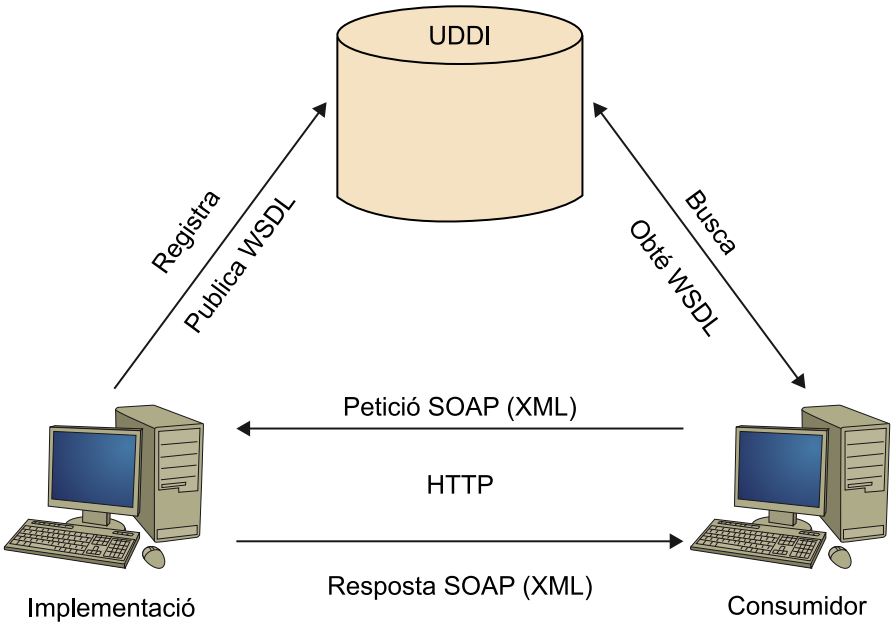
```
<definitions name="HelloService"
   targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>
   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
```

```
        <operation name="sayHello">
           <input message="tns:SayHelloRequest"/>
           <output message="tns:SayHelloResponse"/>
        </operation>
     </portType>

     <binding name="Hello_Binding" type="tns:Hello_PortType">
     <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http"/>
     <operation name="sayHello">
        <soap:operation soapAction="sayHello"/>
        <input>
           <soap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="urn:examples:helloservice"
              use="encoded"/>
        </input>
        <output>
           <soap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="urn:examples:helloservice"
              use="encoded"/>
        </output>
     </operation>
     </binding>

     <service name="Hello_Service">
        <documentation>WSDL File for HelloService</documentation>
        <port binding="tns:Hello_Binding" name="Hello_Port">
        <soap:address
           location="http://www.examples.com/SayHello/">
        </port>
     </service>
  </definitions>
```

UDDI[20] és un element fonamental en el qual es recolzen els serveis web. Aquest estàndard ofereix la possibilitat que els proveïdors o clients de serveis puguin publicar i trobar altres serveis web de tercers. UDDI té un mecanisme perquè els proveïdors de serveis es descriguin a ells mateixos i els diversos serveis que proporcionen, cosa que permet que es puguin registrar i publicar en un registre UDDI. Aquests serveis publicats poden ser cercats, consultats o descoberts per altres usant missatges SOAP. La figura 5 mostra el funcionament d'UDDI.

(20)De l'anglès *universal description discovery and integration*.

Figura 5. Funcionament d'UDDI



Les dades tractades per UDDI es classifiquen en tres categories:

- **Pàgines blanques**: contenen informació general de l'empresa o organització com nom, descripció, informació de contacte, adreça i telèfon.

- **Pàgines grogues**: és molt semblant a l'equivalent telefònic, inclouen categories de catalogació industrial, ubicació geogràfica, etc. Hi ha uns codis i unes claus preestablerts que faciliten la inscripció al registre i així es facilita a tercers la cerca de serveis mitjançant aquests codis de classificació.

- **Pàgines verdes**: contenen informació tècnica sobre un servei web. Normalment inclouen un punter en l'especificació externa i una adreça per a invocar el servei.

SOAP és un protocol d'accés simple a objectes. És una especificació XML per a la creació dels missatges intercanviats entre sistemes distribuïts i la xarxa. Aquest protocol es deriva inicialment de l'XML-RPC. Els missatges estan formats per un sobre (*envelope*), la capçalera (*header*) i el cos (*body*). El sobre embolica el missatge i conté la capçalera i el cos. La capçalera és opcional i només dóna informació per a l'encaminament del missatge.

Exemple

En els documents XML següents, es mostren una petició i una resposta SOAP, respectivament.

Petició SOAP

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
   <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
   </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

Resposta SOAP

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
   <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
   </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

2. Computació en graella

En aquest apartat introduirem els conceptes fonamentals de la computació en graella[21]. Utilitzarem algunes solucions concretes per a exemplificar les característiques de les funcionalitats de la computació en graella. Finalment discutirem els aspectes fonamentals que diferencien els sistemes clàssics d'altes prestacions de la computació en graella. Cal que complementeu els continguts d'aquest apartat amb recerca pròpia d'informació i lectures que us proporcionarem durant el curs.

(21) En anglès, *grid computing*.

2.1. Introducció a la computació en graella

En les darreres dècades i gràcies especialment als avenços en les tecnologies de la informació, la ciència està canviant la manera en la qual es duu a terme. Anteriorment, la contribució científica es podia produir a partir d'un grup relativament petit d'investigadors que treballaven conjuntament en el laboratori. En canvi, els reptes científics actuals requereixen que hi hagi múltiples grups de recerca que col·laborin a escala internacional i comparteixin tota mena d'informació i recursos.

D'aquesta manera, la idea essencial de la computació en graella és la de compartir recursos entre diferents grups de recerca i, en general, entre diferents institucions o organitzacions. En la computació en graella, els recursos estan interconnectats per mitjà de xarxes d'interconnexió de gran abast i utilitzen una capa de programari que permet l'accés transparent als recursos. De fet, el terme *grid* prové del símil entre la idea de compartir recursos distribuïts de manera transparent amb la xarxa elèctrica[22], on els usuaris consumeixen energia de la xarxa sense saber-ne la font o sense saber com s'ha generat (per exemple, si prové de fonts renovables o no). Aquest model és el de proporcionar un servei com a utilitat[23].

(22) En anglès, *power grid*.

(23) En anglès, *utility computing*.

Podem definir la **computació en graella** com una forma de computació distribuïda que ofereix l'abstracció d'un únic computador a partir d'un grup heterogeni de recursos (computadors, instruments, emmagatzemament, etc.) interconnectats per mitjà de xarxes de gran abast (Internet).

Un sistema de computació en graella està compost per múltiples equips informàtics que treballen junts, damunt d'una infraestructura de maquinari i programari, compartint recursos disponibles, però alhora independents entre si. Això vol dir que cada recurs es gestiona de manera autònoma respectant les polítiques locals (per exemple, les polítiques de planificació de treballs).

Malgrat les distàncies físiques a què es poden trobar les màquines que componen la graella, el poder d'unificació i la forma de relació dinàmica d'aquestes màquines aconsegueix respondre a totes les demandes dels usuaris de la xarxa. Dit això, es pot veure que les peticions dels usuaris es distribueixen en diferents tasques per a diferents equips que formen la graella, de tal manera que l'usuari simplement veu les dades finals com un sol còmput.

Cal tenir en compte que l'origen de la computació en graella està íntimament lligat a la compartició de recursos entre la comunitat científica que requereixen altes prestacions per a resoldre problemes complexos. Un exemple clar d'això és el desenvolupament de la computació en graella durant l'última dècada per a poder donar suport a l'LHC[24] al CERN a Ginebra. Una part del repte tecnocientífic és poder emmagatzemar i processar les dades generades en un temps raonable per tal de poder fer descobriments científics de dominis concrets com, per exemple, la física de partícules.

(24)De l'anglès *Large Hadron Collider*.

Un dels grans avantatges de la computació en graella és que es pot muntar sobre la infraestructura de xarxa i de recursos existent, i així aconseguir reduir enormement la despesa inicial. Els recursos de la computació en graella són extremament heterogenis, és a dir, tant pel que fa a maquinari com a programari podem trobar una gran diversitat de plataformes: diferents arquitectures de processadors, sistemes operatius, aplicacions, etc. Això té repercussions importants a l'hora de definir el programari encarregat de gestionar la graella; en primer lloc, la planificació de recursos (decidir sobre quins recursos s'executa una determinada tasca) esdevé un problema força complex; en segon lloc, es fa molt important la utilització d'estàndards de comunicació i d'interfícies que permetin una bona compatibilitat entre programaris tan diferents.

Els recursos de la graella no són dedicats ja que són recursos existents a la institució i possiblement es faran servir per a ús personal o institucional (per exemple, un grup de recerca de la universitat), alhora que per a formar part de la graella. Així doncs, la quantitat de recursos que es fan disponibles per a la graella és variable en funció de la utilització local. Això fa imprescindible incorporar un monitoratge constant en els recursos i polítiques de replanificació amb capacitat per a variar el conjunt de recursos assignats a una tasca en temps d'execució.

La dispersió geogràfica i la gran quantitat de recursos que participen en la graella fa que el nombre d'usuaris potencials sigui molt gran, i amb canvis constants. A més, hi poden haver múltiples institucions involucrades en la graella, cosa que es gestiona a partir del concepte d'*organització virtual*, tal com veurem a continuació.

2.2. Concepte d'*organització virtual*

Foster (2001) va introduir el concepte d'**organització virtual** (OV), que defineix com "una col·lecció dinàmica de múltiples organitzacions que proporcionen flexibilitat, l'intercanvi segur i coordinat de recursos".

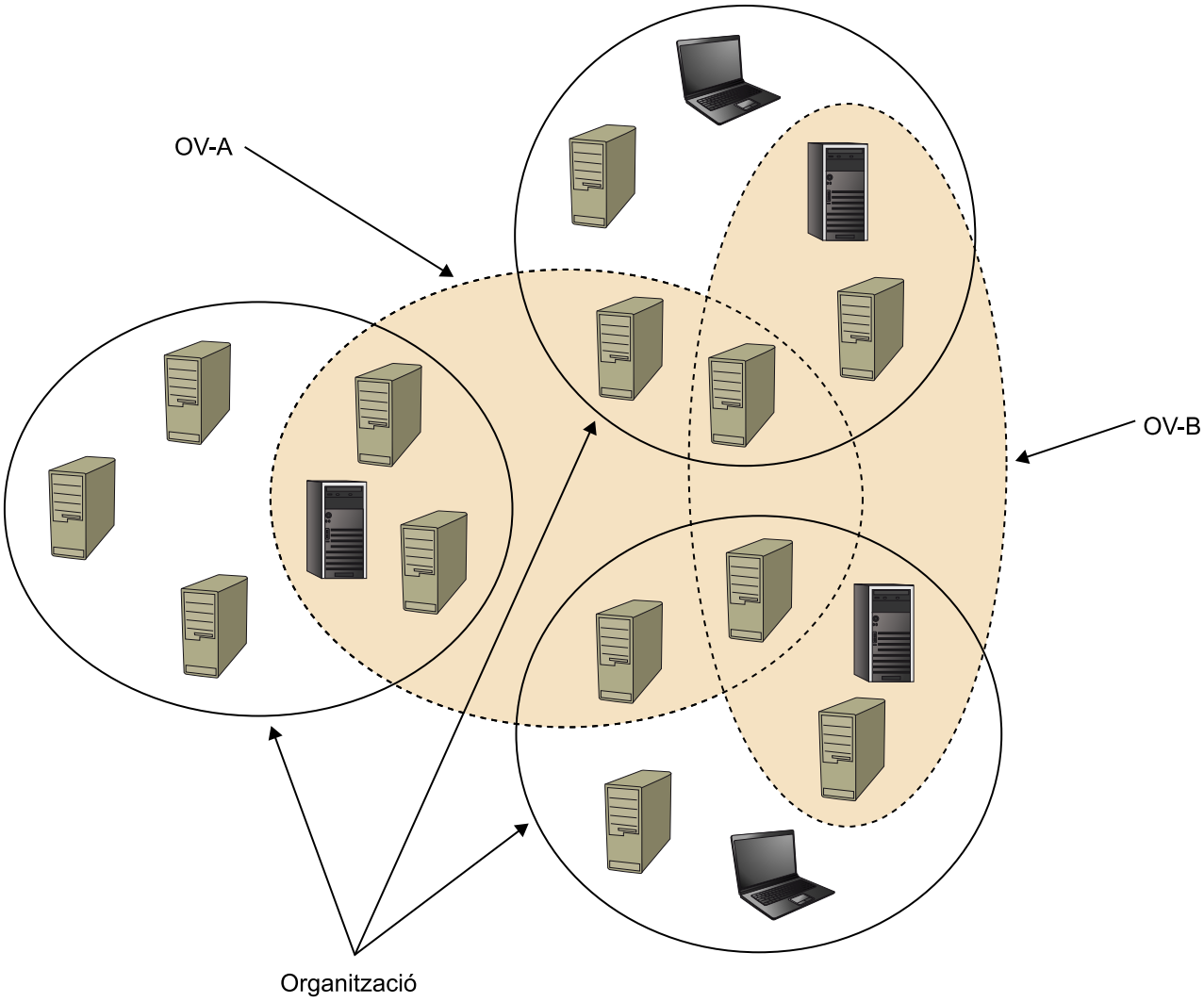
Lectura complementària

I. Foster; C. Kesselman; S. Tuecke (2001). "The Anatomy of the Grid, Enabling Scalable Virtual Organizations". *International Journal of High Performance Computing Applications* (vol. 15, núm. 3).

La figura 6 mostra tres organitzacions reals amb recursos tant computacionals com de dades per a compartir per mitjà de les fronteres organitzacionals. D'altra banda, la figura forma dues organitzacions virtuals, A i B, i cadascuna pot tenir accés a un subconjunt dels recursos en cadascuna de les organitzacions. La virtualització és un mecanisme que millora la utilitat dels sistemes de computació en graella i proporciona als usuaris la possibilitat de configurar el seu entorn.

Cal tenir en compte que per a poder implementar múltiples organitzacions virtuals per sobre d'organitzacions físiques cal utilitzar mecanismes de seguretat i gestió d'usuaris, tal com veurem més endavant. Tot i això, el concepte d'*organització virtual* coincideix clarament amb l'esperit col·laboratiu amb el qual es va iniciar el desenvolupament de la computació en graella.

Figura 6. Exemple de dues organitzacions virtuals



2.3. El programari intermediari

De tot el control i gestió de la xarxa se n'encarrega el programari intermediari (*middleware*), el programari instal·lat en els recursos de la graella que hi ha entre el sistema operatiu i les aplicacions que utilitzen la graella. Hi ha una sèrie de qüestions comunes, com, per exemple, els esquemes de seguretat (autenticació d'usuaris i recursos), que s'acostumen a implementar en tots els programaris intermediaris, però habitualment hi ha recursos a la xarxa destinats a la gestió de la graella, els anomenats *gestors* (*brokers*).

El programari intermediari dels recursos d'una graella també té per objectiu fer altres tasques, com, per exemple, la replicació de les dades i les metadades que requeriran els nodes que executin una certa tasca per qüestions d'eficiència o tolerància a fallades. Per tal de simplificar aquest tipus de tasques es van definir una sèrie de protocols especialitzats en xarxes en graella, com, per exemple, el GridFTP, el qual veurem amb detall més endavant.

En aquest subapartat ens centrarem en el Globus Toolkit com a exemple de programari intermediari, ja que es va convertir en l'estàndard *de facto*. Tot i això cal tenir en compte que hi ha altres programaris intermediaris per a la computació en graella, com, per exemple, Unicore, Condor i gLite.

Activitat

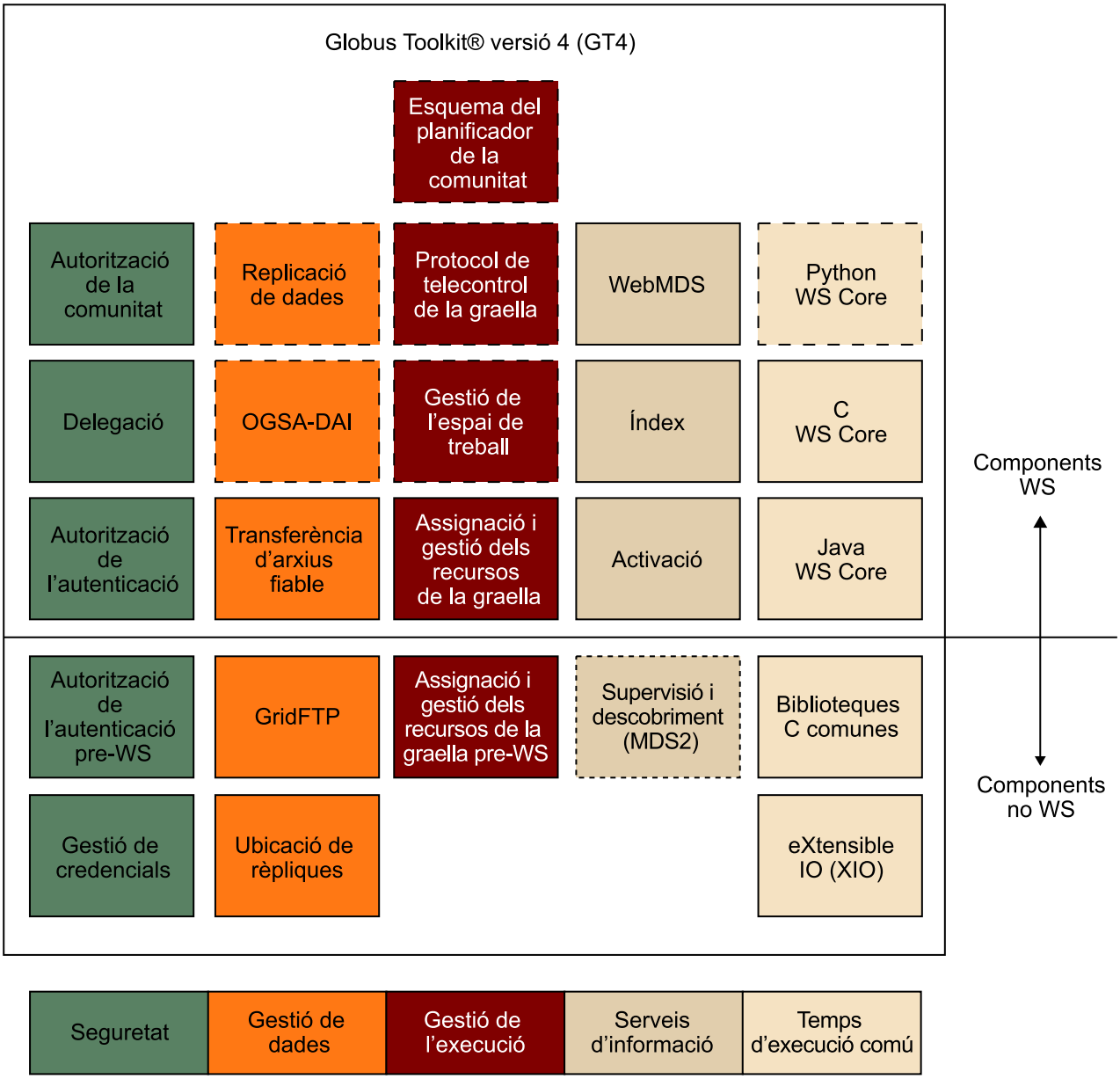
Busqueu informació addicional sobre diferents programaris intermediaris per a computació en graella i analitzeu-ne els pros i els contres.

La Globus Alliance és el nom d'un esforç internacional entre diferents institucions i individus que té per objectius la recerca i el desenvolupament de tecnologies i estàndards per a la computació en graella. Una de les seves tasques principals és la construcció del Globus Toolkit, un conjunt de solucions per a problemes típics de la computació en graella com ara mantenir la seguretat en entorns distribuïts i heterogenis, gestionar recursos, monitorar, solucionar fallades i gestionar dades, que permeten construir programari intermediari per cobrir les necessitats de gestió de la graella. El desenvolupament del Globus Toolkit ha tingut des del principi un enfocament de programari lliure i una construcció descentralitzada, i també una clara preferència per l'adopció d'estàndards. També cal destacar l'ús de serveis web pel que fa a interfície entre mòduls, basada en mecanismes XML, que aconsegueix un sistema distribuït poc acoblat[25].

(25) En anglès, *loosely coupled*.

L'arquitectura del Globus Toolkit es pot diferenciar en cinc pilars bàsics, com podem veure en la figura 7. Per una banda, els quatre fonaments de la gestió de la graella en forma de biblioteques, incloent-hi seguretat, gestió de dades, gestió de l'execució i serveis d'informació. Per l'altra, el Globus Toolkit ofereix interfícies per a diferents llenguatges de programació (C, Java i Python) que permeten desenvolupar serveis web basats en la resta de serveis oferts pel Globus Toolkit. Un darrer grup de components oferts pel Globus Toolkit són les implementacions de serveis (aquells per sota de la línia negra a la figura, etiquetats com "Components no WS"). Com indica el nom, són implementacions de solucions pròpies de la graella (per exemple, GridFTP i GRAM) sense estar basades en serveis web.

Figura 7. Diagrama dels mòduls que formen el Globus Toolkit



Component Core GT: interfícies públiques congelades entre versions incrementals.

Contribució/previsualització tècnica: les interfícies públiques poden canviar entre versions incrementals.

Component obsolet: no s'accepta; s'eliminarà en una versió futura.

2.3.1. Gestió de l'execució

El principal servei ofert pel Globus Toolkit en matèria de gestió de l'execució és el servei de gestió i repartiment de recursos de la graella (GRAM[26]), que proporciona una interfície per a enviar, monitorar i cancel·lar tasques. D'aquesta manera, unes terceres parts es poden basar en aquestes implementacions, per exemple per a construir planificadors de tasques.

(26)De l'anglès *grid resource allocation and management*.

Lectura complementària

Podeu trobar molta més informació referent a l'arquitectura interna d'aquest mòdul en el document "Globus Toolkit 4.0 WS GRAM Approach".

2.3.2. Gestió de les dades

Dins el món de la computació en graella, la transferència de dades és un problema fonamental, perquè ens trobem amb conjunts de nodes computacionals repartits per localitzacions geogràfiques molt disperses. A més, hi ha certs tipus d'aplicacions que necessiten un ús intensiu de dades, com és el cas de l'LHC construït al CERN.

El Globus Toolkit ofereix distintes peces de programari que donen solució a alguns dels problemes relacionats amb la transferència i l'accés a grans quantitats de dades. Per una banda, la implementació de l'especificació GridFTP proporciona eines per a fer transferències segures i d'alt rendiment de memòria a memòria o de disc a disc, que a més té compatibilitat amb clients i servidors FTP tradicionals. Per l'altra, el servei de replicació d'ubicacions[27] proporciona un servei descentralitzat per a mantenir informació sobre la ubicació de conjunts de dades i fitxers repartits per la xarxa. Una altra solució, com el servei de transferència fiable de fitxers[28], proporciona una nova capa sobre el GridFTP que permet fer transferències múltiples i fiables de fitxers a gran escala.

(27)En anglès, *replica location service*.

(28)En anglès, *reliable file transfer*.

2.3.3. Serveis d'informació

Quant a monitorització i descobriment dels recursos en graella, el Globus Toolkit ofereix eines per a convertir les dades relacionades sobre els recursos en entitats XML que descriuen les característiques d'aquests recursos. Aquests serveis, basats en les especificacions WSFR i WS-Notification, es complementen amb altres d'indexació, els quals no estan basats en els anteriors estàndards (per a aquells nodes que utilitzin un sistema en què aquests no estiguin implementats).

2.3.4. Seguretat

Les propietats dels entorns de computació en graella, principalment la dispersió geogràfica d'usuaris i els recursos de diferents institucions, organismes, entitats, etc., fan de la seguretat un tema important. Establir la identitat d'usuaris i recursos, definir i aplicar polítiques de seguretat, reforçar la seguretat de les comunicacions, són els pilars bàsics d'una política de seguretat de la computació en graella. En el Globus Toolkit hi trobem components de seguretat de baix nivell que implementen protocols per a la protecció de missatges, auten-

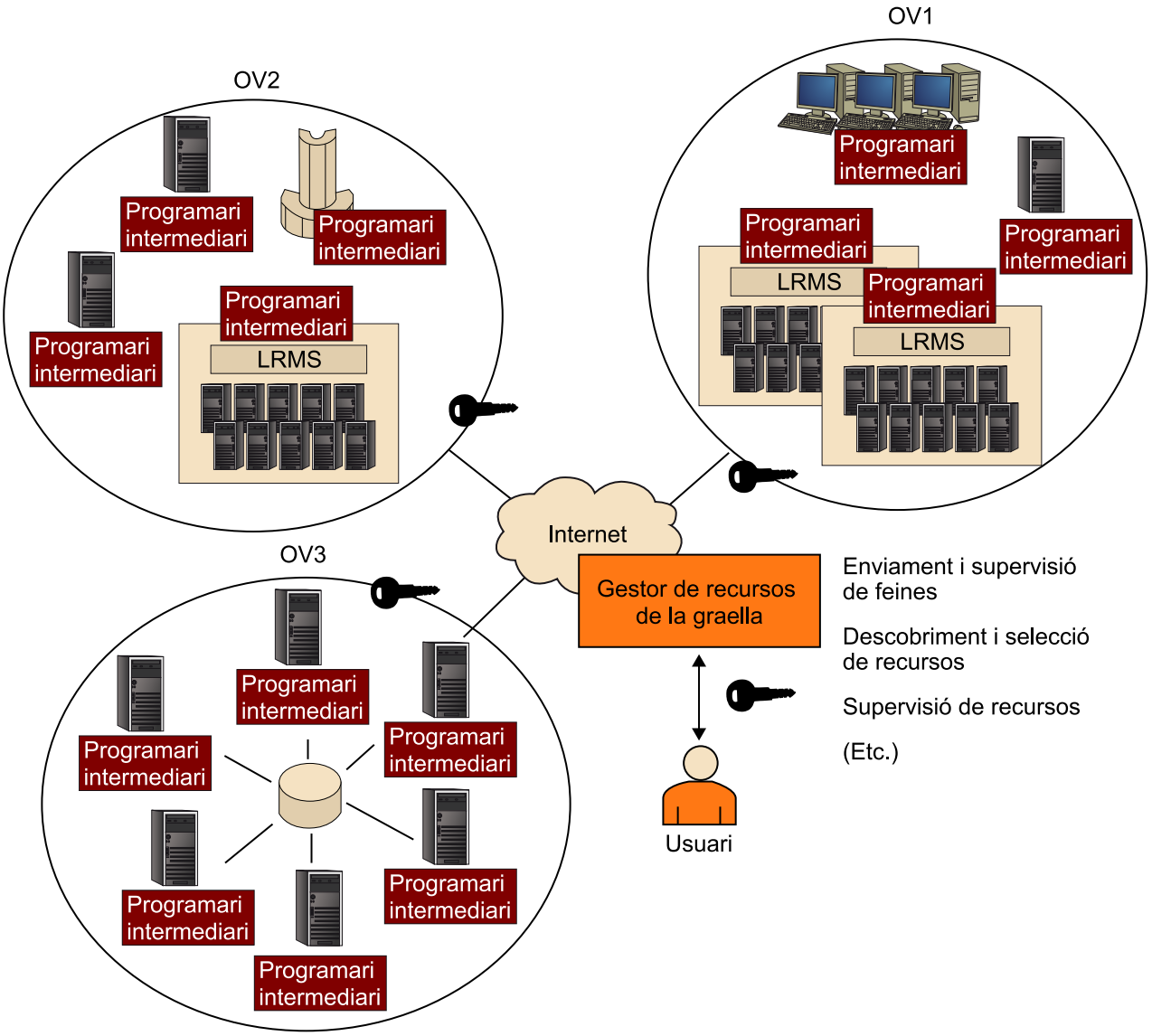
ticació, delegació i autorització. Com es mostra en la figura 7, es proporciona suport per a solucions que compleixen l'estàndard WS-Security amb credencials X.509, i també com amb usuaris i contrasenyes, i per a seguretat pel que fa a transport amb credencials X.509. En la configuració per defecte del Globus Toolkit 4, cada usuari i cada recurs té una clau pública X.509 que permet als diferents protocols fer la validació entre dues entitats, i també establir-hi un canal segur de comunicació.

2.4. Metaplanificació

En la computació en graella, les tasques són aplicacions preparades per a executar-les, empaquetades juntament amb les dades d'entrada necessàries i la descripció de les seves necessitats tecnològiques. Si el programari intermediari segueix els estàndards, utilitzem el *job submission description language*, una especificació d'XML que ens permet definir aspectes com els requeriments que han de tenir els computadors per a poder executar-la (memòria RAM disponible, SWAP disponible, CPU, sistema operatiu, etc.), variables d'entorn necessàries per a executar-la correctament, i també les dades de què depèn, etc.

Una de les funcions més importants de la gestió de la graella consisteix a decidir quins seran els recursos que utilitzarà una determinada tasca. Atesa la diversitat de recursos que formen la graella, i també l'heterogeneïtat (diferents sistemes operatius, arquitectures, programari, etc.) i el dinamisme que tenen, aquesta feina requereix uns algoritmes de planificació més propis d'un sistema operatiu distribuït. Aquesta és la funció del metaplanificador o gestor (*meta-scheduler* o *broker*, respectivament), una part del programari intermediari que s'ocupa de monitorar els nodes i de planificar l'execució de les tasques sobre aquests nodes de la manera més efectiva possible. Habitualment també disposa de mètodes de recuperació de fallades, i per a evitar els punts únics de fallada, en molts casos estan distribuïts entre més d'un node de la xarxa o treballen de manera redundant. En el cas de la figura 8 el metaplanificador (*gestor de recursos de la graella*) és centralitzat.

Figura 8. Arquitectura en graella amb un metaplanificador centralitzat



Tot i que hi ha desenes de metaplanificadors per a sistemes de computació en graella, en aquest subapartat veurem els aspectes més generals de GridWay, que és un dels més coneguts.

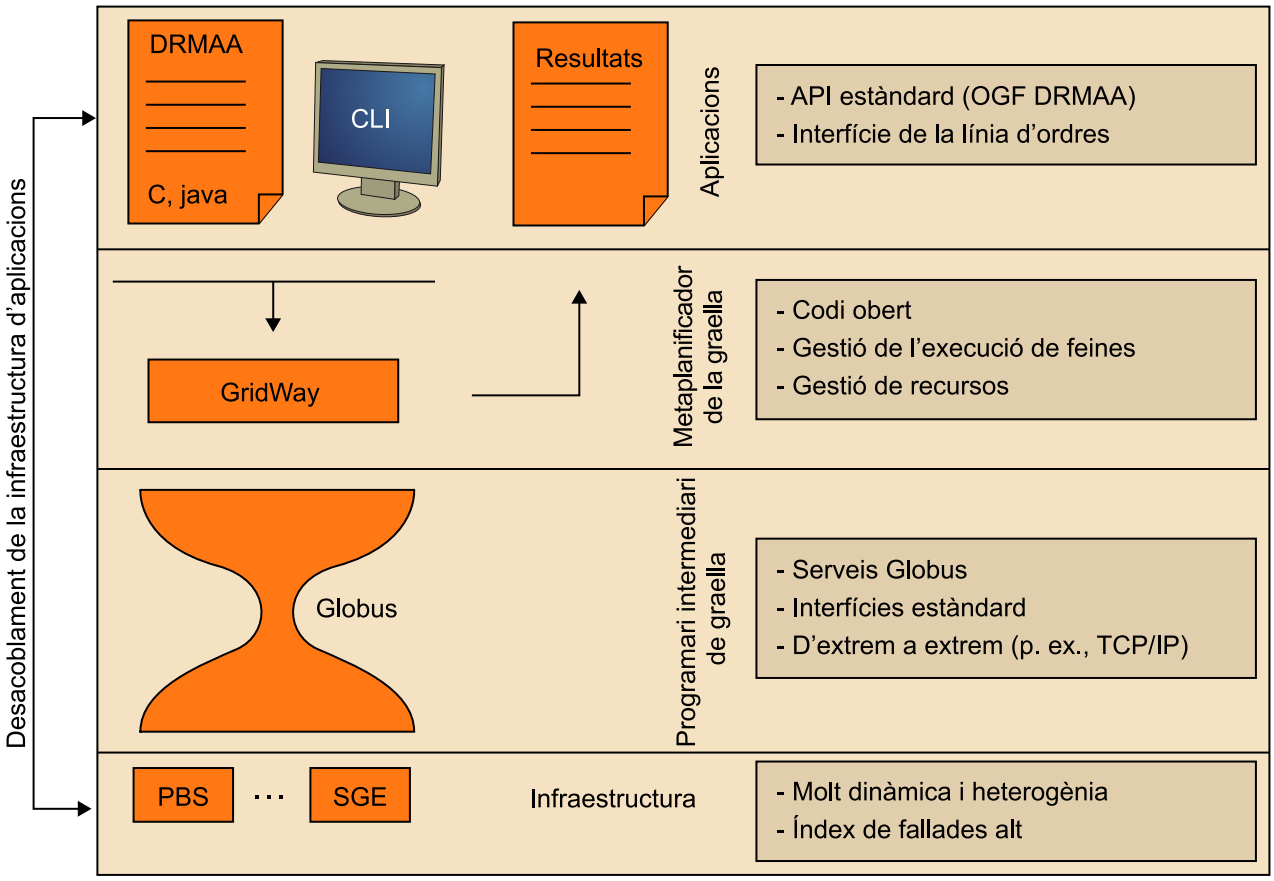
Activitat

Busqueu informació addicional sobre diferents metaplanificadors per a computació en graella i estudieu-ne les característiques. També podeu buscar articles resum sobre metaplanificadors en graella.

GridWay és una solució desenvolupada a la Universitat Complutense de Madrid. Aquest metaplanificador, llicenciat com a programari lliure i creat amb la filosofia de la Globus Alliance, està pensat per a funcionar amb serveis de Globus, com el Globus Toolkit.

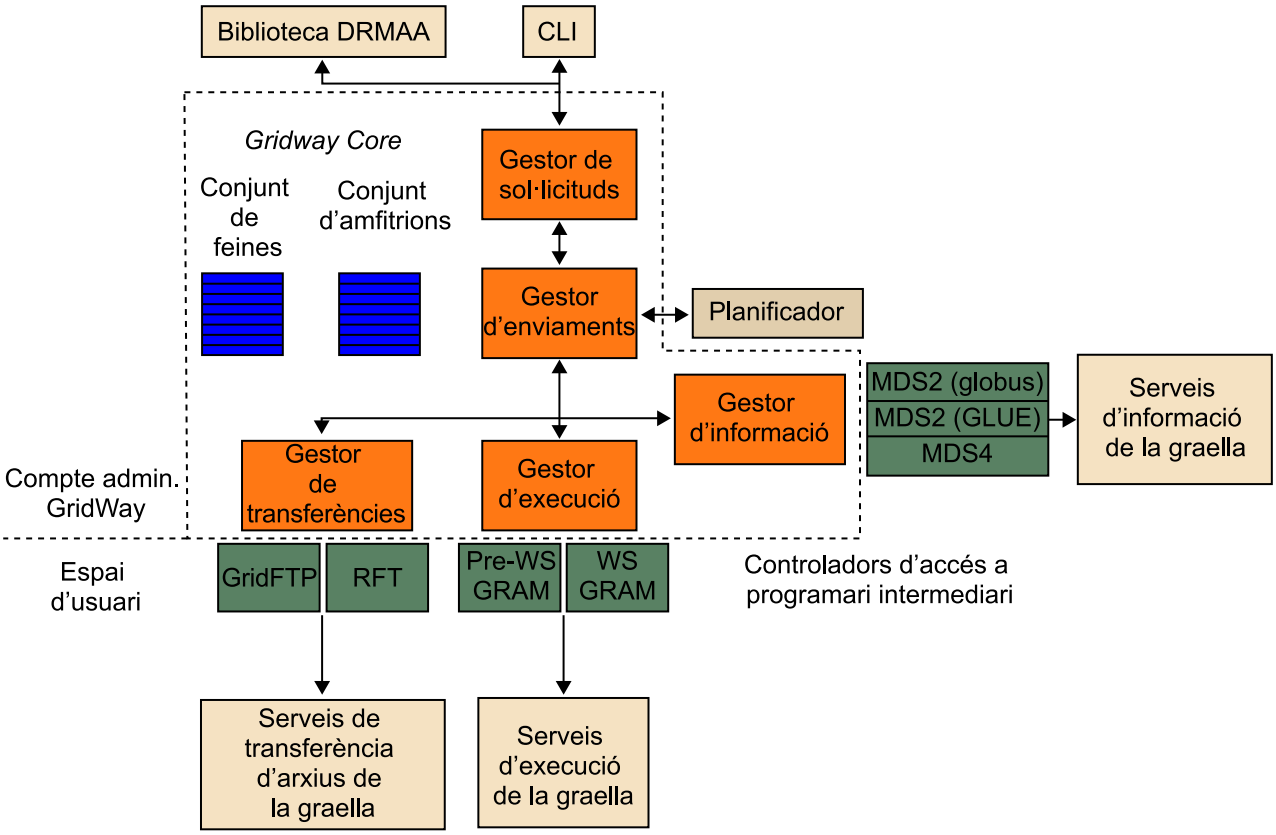
La figura 9 mostra el lloc que ocupa el metaplanificador GridWay en la infraestructura de la graella, i virtualitza un servei de planificació i aconsegueix separar la infraestructura de les aplicacions que es poden executar sobre la graella.

Figura 9. Capes que formen una xarxa en graella utilitzant GridWay



Tal com es mostra en la figura 10, GridWay proporciona una interfície d'usuari basada en línia d'ordres, però també permet utilitzar l'especificació Distributed Resource Management Application (DRMAA) establerta per l'Open Grid Forum, que té per objectiu que les aplicacions mateixes puguin interactuar amb la graella directament.

Figura 10. Diagrama de l'arquitectura de GridWay



GridWay ens ofereix una gran quantitat de polítiques de planificació, algunes dels quals descriurem breument a continuació. Veurem aquestes polítiques a manera d'exemple tot i que n'hi ha moltes de diversos tipus amb diferents objectius d'optimització tant per a GridWay com per altres metaplanificadors. La tasca de la política de planificació és, com indica el nom, decidir l'ordre d'execució de les tasques al Grid.

Activitat

Busqueu informació addicional sobre polítiques de planificació per a metaplanificadors per a computació en graella. També podeu buscar articles resum sobre polítiques de metaplanificadors en graella.

A continuació enumerarem tres polítiques basades en prioritats estàtiques en les tasques d'un usuari o d'un grup en concret, o *fixed priority policy* (FP). El rang va de 00 a 19 i l'enviament de tasques a la graella prioritza les de valor inferior:

1) La política *fair-share* permet establir unes ràtios de prioritat per a l'enviament de tasques a la graella per als diferents usuaris. Per exemple, podem establir una ràtio 10:4 per a les tasques d'un grup de recerca o departament determinat. El problema d'aquesta política és que no té en compte l'ús dels recursos, un aspecte fonamental a l'hora d'aprofitar de manera adequada un sistema de computació en graella.

2) La política *waiting-time* té per objectiu evitar que les tasques amb una prioritat inferior no passin mai a executar-se perquè d'altres de més prioritat monopolitzen els recursos contínuament (problema d'assignació de recursos compartits conegut en el món de la computació com a *inanició*). El que fa és incrementar linealment la prioritat de les tasques a mesura que passa el temps en què espera per a ser executada.

3) Els límits temporals d'execució[29] es poden marcar gràcies a la política *deadline*, que incrementa la prioritat d'execució d'una tasca a mesura que s'acosta. Tot i això és important notar que no parlem d'un planificador de temps real, perquè aquesta política no assegura que les tasques s'executin abans del temps especificat.

(29)En anglès, *deadlines*.

A part d'aquestes polítiques basades en prioritat val la pena destacar el mecanisme anomenat *matchmaking*, que és utilitzat en moltes polítiques i es va desenvolupar en el context de Condor. En poques paraules, *matchmaking* s'encarrega de fer un aparellament dels treballs amb els recursos de manera que se satisfan els requeriments d'aquests recursos.

Activitat

Busqueu informació sobre tècniques de *matchmaking* per a la computació en graella. També busqueu informació sobre la plataforma Open Science Grid.

2.5. Estandardització

L'Open Grid Forum (OGF) és una comunitat d'usuaris, desenvolupadors i proveïdors per a l'estandardització de la computació en graella. Es va formar el 2006 com a fusió del Global Grid Forum i de l'Enterprise Grid Alliance. L'OGF té dues funcions principals (a part de les funcions administratives):

- Elaboració de normes relatives a les xarxes.
- Construcció de comunitats dins de la xarxa global (incloent-hi l'ampliació per a arribar a una participació més àmplia de l'acadèmia i la indústria).

Cadascuna d'aquestes àrees funcionals es divideix en grups de tres tipus:

- Grups de treball amb un paper ben definit (en general produeix un estàndard).
- Grups de recerca amb un paper més fluix que reuneix la gent per discutir desenvolupaments dins del seu camp i generar casos d'ús i fer grups de treball.
- Grups de la comunitat.

Les principals normes que han estat produïdes per l'OGF són, entre moltes altres, les següents:

- GridFTP: extensions en el protocol de transferència d'arxius per a la transferència de dades a alta velocitat, segura i fiable.

- Grid Laboratory Uniform Environment (GLUE): model d'informació independent de la tecnologia per a una representació uniforme dels recursos en graella.

- Single API for Grid Applications (SAGA): especificació d'alt nivell per a aplicacions en xarxa que descriu una interfície de programació d'aplicacions en graella.

- Open Grid Services Architecture (OGSA): norma que descriu una arquitectura per a un servei orientat a l'entorn informàtic de xarxa per a ús empresarial i científic.

- Distributed Resource Management Application API (DRMAA): especificació API d'alt nivell per al control i presentació dels treballs a un o més sistemes distribuïts de gestió de recursos (DRM) dins d'una arquitectura Grid.

- Enviament de treballs de descripció del llenguatge: especificació extensible XML per a la descripció de les tasques simples per no interactives sistemes informàtics d'execució. L'especificació se centra en la descripció de les propostes de tasques computacionals en els tradicionals sistemes de computació d'alt rendiment, com programadors de lots.

2.6. Computació en graella enfront de supercomputació

El que realment estableix una barrera qualitativa entre la computació en graella i la supercomputació és l'heterogeneïtat. Els nodes que conformen una graella poden ser pràcticament qualsevol cosa, des d'un ordinador de taula fins a un sensor de l'accelerador de partícules del CERN. A més, a dins de la mateixa graella hi poden conviure diferents sistemes operatius i diferents arquitectures: des dels supercomputadors dedicats que funcionen amb un sistema del tipus Unix fins a les agendes personals amb Windows Mobile o Android de l'equip directiu de l'empresa. Els supercomputadors (que són clústers per definició), en canvi, es componen de maquinari de característiques similars i el mateix sistema operatiu.

D'altra banda, els nodes d'un clúster estan altament acoblats, situats al mateix domini de xarxa i habitualment connectats amb interfícies de xarxa d'altes prestacions. A més, els nodes són dedicats i, per tant, executen únicament les aplicacions del clúster. En aquestes condicions (connexions ràpides, nodes dedicats, alt acoblament) el més habitual és utilitzar MPI per a programar apli-

cacions. D'aquesta manera, els clústers són ideals per a executar aplicacions subdividides en molts problemes petits i amb una gran quantitat de comunicació entre processos, les aplicacions paral·leles de gra fi.

En canvi, en la computació en graella, els nodes solen estar distribuïts entre diferents subxarxes i dominis, ja que estan dispersos entre diferents organitzacions. A més, és un entorn dinàmic al qual podem afegir o del qual podem eliminar nodes en qualsevol moment, i a més els nodes no són completament dedicats, amb el consegüent dinamisme a l'hora de repartir les tasques. L'ús de màquina de xarxa de gamma baixa (aprofitant els preus de l'economia d'escala) i la dispersió geogràfica dels seus nodes, les aplicacions més a apropiades per a ser executades en un Grid són les *coarse-grained*, aquelles en què la comunicació entre processos té poca rellevància enfront de la quantitat de computació necessària. Cal dir, però, que també hi ha eines per a MPI que funcionen amb computació en graella com ara MPICH-G2.

També cal dir que tot i que habitualment la computació en graella utilitza maquinari de gran consum, també es poden afegir computadors d'altes prestacions a la graella i així sumar potència de càlcul o simplement donar accés a computació d'altes prestacions que només està disponible en altres institucions.

3. Computació en núvol

En aquest apartat estudiarem els fonaments de la computació en núvol. Primer ens centrarem en els conceptes bàsics i aspectes generals de la computació en núvol, per a centrar-nos després en els aspectes relacionats amb la computació d'altes prestacions. Cal tenir en compte que si bé la computació en graella va aparèixer com a mitjà per a poder millorar la col·laboració científica i així poder resoldre problemes de gran rellevància, la computació en núvol reutilitza gran part de l'experiència obtinguda amb la computació en graella però té un component principalment econòmic, basat en un model de mercat.

3.1. Introducció a la computació en núvol

La computació en núvol[30] es presenta com la materialització més propera a la computació com a utilitat[31]. Actualment veiem una gran transformació de la indústria de les tecnologies de la informació a causa del model de computació en núvol, el qual està basat en la idea de proveir programari com a servei. Aquest model també està influenciant directament com es dissenyen, s'utilitzen i es compren el maquinari i els sistemes de tecnologies de la informació. Per exemple, el model de computació en núvol permet que una petita empresa pugui començar a operar sense haver de fer grans inversions en maquinari i els serveis associats per a operar-hi ja que pot comprar els serveis disponibles en el núvol només quan els necessiti. A més, l'associativitat del cost de la computació en núvol fa que es puguin assolir solucions escalables sense cost addicional. Per exemple, utilitzar 1.000 servidors durant una hora costa el mateix que utilitzar-ne un durant 1.000 hores. Aquest model basat en l'elasticitat dels recursos fa que la computació en núvol tingui un gran potencial per a revolucionar la computació en molt àmbits.

(30) En anglès, *cloud computing*.

(31) En anglès, *utility computing*.

Una de les definicions més conegudes de la computació en núvol és la proporcionada per l'Institut Nacional d'Estàndards i Tecnologia (NIST) del Departament de Comerç dels Estats Units d'Amèrica, que presentem a continuació.

> La **computació en núvol** és un model que possibilita l'accés ubic, convenient, per encàrrec a un conjunt de recursos configurables en xarxa (per exemple, xarxes, servidors, emmagatzematge, aplicacions i serveis) que es poden proveir ràpidament i alliberar amb un mínim esforç de gestió o d'interacció del proveïdor de serveis.

Aquest model en núvol està compost de cinc característiques essencials, tres models de servei i quatre models de desplegament.

Les característiques essencials del model en núvol són les següents:

1) Autoservei per encàrrec. Un consumidor pot ser proveït unilateralment amb recursos computacionals com ara temps de servidor i emmagatzematge en xarxa, de manera automàtica i a mesura que ho necessita sense que li calgui interacció humana de cada proveïdor de serveis.

2)Ampli accés a la xarxa. Els recursos estan disponibles a la xarxa i s'hi pot accedir pels mecanismes estàndard que promouen l'ús de plataformes heterogènies de clients lleugers o normals (per exemple, telèfons mòbils, tauletes, portàtils i estacions de treball).

3)Agrupació de recursos. Els proveïdors dels recursos computacionals s'agrupen per servir a múltiples consumidors utilitzant un model multillogater, amb diferents recursos físics i virtuals assignats i reassignats de manera dinàmica d'acord amb la demanda del consumidor. Hi ha un sentit d'independència de la ubicació en què el client generalment no té cap control o coneixement sobre la ubicació exacta dels recursos proporcionats, però pot ser capaç d'especificar la ubicació a un nivell més alt d'abstracció (per exemple, país, estat o centre de dades). Alguns dels exemples de recursos són l'emmagatzematge, el processament, la memòria i l'amplada de banda de la xarxa.

4) Ràpida elasticitat. Els recursos poden ser proveïts i alliberats elàsticament, en alguns casos de manera automàtica, per a escalar ràpidament tant per a créixer com decréixer en funció de la demanda. Des del punt de vista del consumidor, els recursos disponibles sovint semblen il·limitats i poden ser assignats en qualsevol quantitat en qualsevol moment.

5) Servei mesurat. Els sistemes en núvol controlen i optimitzen automàticament l'ús dels recursos mitjançant la capacitat de monitorar en un cert nivell d'abstracció adequat per al tipus de servei (per exemple, emmagatzematge, processament, amplada de banda i comptes d'usuari actius). L'ús de recursos pot ser monitorat, controlat i reportat, de manera que proporciona transparència tant per al proveïdor com per al consumidor del servei utilitzat.

La definició del NIST també inclou tres models de servei (SaaS, PaaS, IaaS) i quatre models de desplegament o tipus de núvols (privat, de comunitat, públic, híbrid), que veurem més endavant.

Des del punt de vista del maquinari hi ha tres aspectes nous en la computació en núvol:

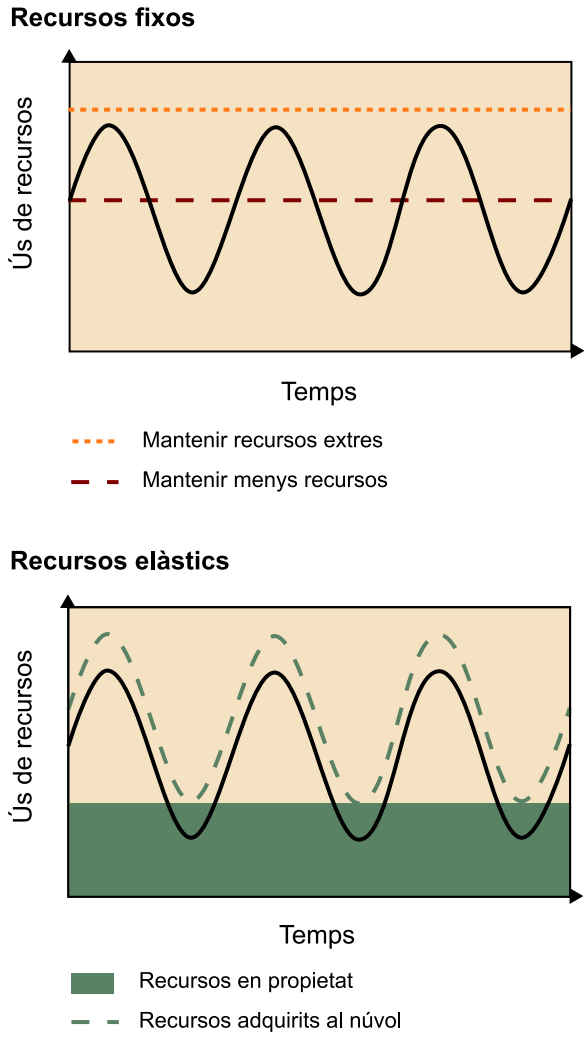
- La il·lusió d'infinits recursos informàtics disponibles per encàrrec, que elimina la necessitat dels usuaris del núvol de planejar amb anticipació la provisió de recursos necessaris.

- L'eliminació d'un compromís per avançat dels usuaris del núvol, que permet a les empreses començar a poc a poc i augmentar els recursos de maquinari només quan hi ha un augment en les seves necessitats.

- La possibilitat de pagar per l'ús dels recursos informàtics (model *pay as you go*) a curt termini, segons que sigui necessari (per exemple, els processadors per hora i l'emmagatzematge per dia) i alliberar-los quan sigui necessari, de manera que s'alliberen màquines i emmagatzematge quan ja no són útils.

Qualsevol aplicació necessita un model de càlcul, un model d'emmagatzematge, i un model de comunicació. La multiplexació estadística necessària per a aconseguir l'elasticitat i la il·lusió d'infinita capacitat requereix que els recursos siguin virtualitzats per a ocultar la implementació de la manera com es multiplexen i es comparteix. Més endavant estudiarem aquests mecanismes de virtualització i les solucions més populars en la computació en núvol.

Figura 11. Provisió de recursos en núvol estàtica enfront d'elàstica



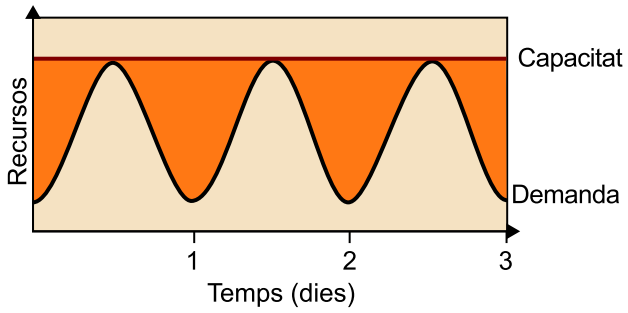
Encara que l'atractiu econòmic de la computació en núvol és sovint descrit com la conversió de les despeses de capital en despeses d'explotació, el model de pagament per ús capta més bé el benefici econòmic per al comprador. La idea bàsica és que l'absència de despeses de capital per avançat permet que el capital es pugui invertir en negoci. Tot i això, pot succeir que pagar per ús de recursos en núvol sembli tenir un preu similar al de comprar els recursos però el núvol permet transferir el risc als proveïdors, especialment els riscos d'excés d'aprovisionament (subutilització) i de manca d'aprovisionament (saturació).

L'elasticitat dels recursos és una de les claus ja que permet que els usuaris puguin adaptar els recursos utilitzats (i pels quals paguen) a la demanda, sia perquè la utilització dels serveis té fluctuacions periòdiques (per exemple, més clients durant el dia que durant la nit) o també per alguns esclats inesperats de la demanda a causa d'esdeveniments externs (per exemple, els esdeveniments de notícies), tal com il·lustra la figura 11.

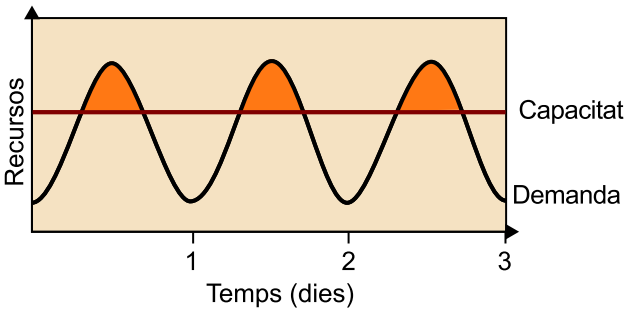
Hem de tenir en compte que adquirir nous equips pot requerir unes quantes setmanes i l'única manera de resistir aquests pics en la demanda és el subministrament anticipat. Això fa que si es disposa dels recursos per a resistir els pics de demanda es perdi capacitat, tal com mostra la figura 12a. També pot succeir que se subestimi el pic de demanda (figura 12b) i no es disposi de prou recursos per a donar servei als clients. Mentre que els efectes monetaris d'excés d'aprovisionament són fàcils de mesurar, els de falta d'aprovisionament són més difícils de mesurar però potencialment igual de greus: no solament es rebutgen usuaris, de manera que es generen zero ingressos, sinó que hi haurà molts usuaris que es perdran per sempre per un mal servei. La figura 12c té com a objectiu captar aquest comportament: els usuaris abandonen un servei que està saturat fins que la càrrega pic és igual a la capacitat d'ús del centre de dades, en què els usuaris tornen a rebre un punt de servei acceptable, però amb un nombre inferior d'usuaris potencials.

Figura 12. Exemplificació de diversos escenaris de provisió de recursos

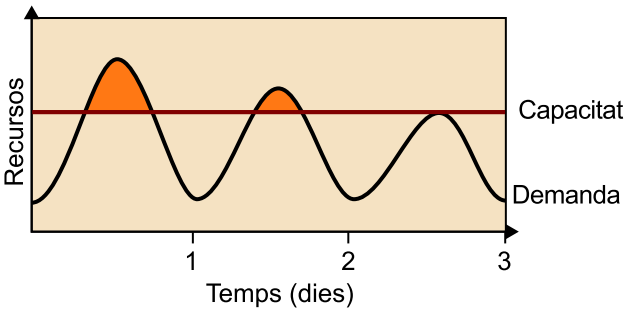
a. Aprovisionament per pic de càrrega



b. Aprovisionament insuficient 1



c. Aprovisionament insuficient 2



Un dels pioners de la computació en núvol va ser Salesforce.com, que el 1999 va introduir el concepte d'*entrega d'aplicacions empresarials* per mitjà d'una web bàsica. Amazon va ser la següent i va llançar Amazon Web Service (2002). Aleshores va aparèixer Google Docs (2006), que va portar la computació en núvol a l'avantguarda de la consciència del públic. També Elastic Compute Cloud d'Amazon (Amazon EC2) va sorgir el mateix any com un servei web comercial que va permetre a les empreses i els particulars llogar equips en què es poguessin executar les seves pròpies aplicacions informàtiques. El 2008 va sorgir Eucalyptus, la primera plataforma de codi obert compatible amb les interfícies d'Amazon Web Services per a poder enllaçar els núvols privats amb el proveïdor de serveis en núvol més popular. Una mica més tard va aparèixer OpenNebula, el primer programari de codi obert per a la implementació dels núvols privats i híbrids. Windows Azure, la plataforma en núvol de Microsoft es va donar a conèixer el 2009. El 2010, les plataformes adopten una

nova estructura pròpia amb les evolucions de diferents capes de servei: client, aplicació, plataforma, infraestructura i servidor. El 2011, Apple es va unir al club de proveïdors de serveis en núvol amb el seu servei iCloud, un sistema d'emmagatzematge en el núvol orientat a música, vídeos, fotografies, aplicacions i calendaris.

3.2. Característiques de la computació en núvol

En aquest subapartat ens centrarem en les característiques generals de la computació en núvol, les quals estan orientades a entorns empresarials i models de negoci i no tant a computació d'altes prestacions. Més endavant estudiarem les implicacions i les possibilitats de la computació en núvol per a la computació d'altes prestacions.

El model de computació en núvol té una sèrie de **característiques** que es poden veure com a avantatges comuns per a la majoria d'empreses, com ara:

- **Flexibilitat**: les empreses poden contractar el que necessiten en cada moment ja que els requeriments d'una empresa poden dependre de molts factors i per tant ser molt variables. En un sistema tradicional si, per exemple, es requeria un sistema amb molta capacitat de computació durant un període reduït no se'n podien amortitzar el costos.

- **Cost**: el model de pagament per ús permet a les companyies reduir els costos fixos i les inversions en tecnologies de la informació, a més d'aprofitar al màxim els seus diners ja que poden pagar exactament per allò que necessiten. El preu d'aquest tipus de serveis es competitiu a causa de l'explotació de les economies d'escala.

- **Subcontractació**: el núvol permet externalitzar responsabilitats relacionades amb la gestió de les tecnologies de la informació, i així reduir el personal que es necessita per a dur a terme aquestes tasques.

- **Gestió de versions**: el client no s'ha de preocupar d'actualitzar la seva infraestructura o de comprar llicències de programari més noves. El proveïdor del servei s'encarrega de l'actualització de les llicències amb els costos que hi estan associats.

- **Seguretat**: el proveïdor és l'encarregat de garantir la seguretat del sistema i les dades evitant que elements externs o altres usuaris hi puguin accedir, encara que físicament estiguin usant els mateixos servidors. A més, es poden oferir serveis de còpies de seguretat de manera transparent per a evitar les pèrdues de dades en cas de fallada.

- **Disponibilitat**: el proveïdor ha de garantir que el servei estigui en funcionament durant el temps contractat. Aquest aspecte és crític ja que l'activitat d'una empresa no es pot aturar per problemes informàtics, de

manera que per a garantir això amb el sistema tradicional caldria tenir duplicitat dels elements del sistema.

Aquestes característiques del model de computació en núvol tenen una sèrie d'**avantatges** respecte als models tradicionals de computació, com ara:

- La presentació dels serveis a escala global: gràcies a la replicació de serveis a diferents llocs del món es pot fer accessible la informació amb molt bona qualitat de treball.

- Les infraestructures de computació en núvol proporcionen una més bona capacitat d'adaptació, recuperació completa de pèrdua de dades mitjançant les còpies de seguretat i accés a la informació en qualsevol moment.

- A un usuari que utilitza una infraestructura cent per cent de computació en núvol no li cal preocupar-se del maquinari que hi ha ni de fer-ne el manteniment, perquè tot això passa a mans del proveïdor del servei, fet que requereix molta menys inversió en maquinari i programari i permet començar a fer feina més aviat.

- Les implementacions de les aplicacions solen ser molt més ràpides, ja que n'hi ha moltes que estan fetes per mitjà de sistemes base que permeten un cert nivell de personalització.

- Les actualitzacions del programari no cal fer-les a les màquines físiques sinó que es fan automàticament al núvol. Així doncs, no es perd temps en actualitzacions.

- La computació en núvol ajuda a disminuir el consum energètic, ja que els centres de dades de les empreses no han de mantenir els seus servidors arrancats perquè tot està externalitzat.

La computació en núvol també té una sèrie d'**inconvenients** que cal tenir en compte abans d'adoptar solucions en núvol, com ara:

- La centralització de les aplicacions i l'emmagatzematge de dades origina una dependència amb el proveïdor de serveis.

- La disponibilitat dels serveis i de les dades està lligada a tenir accés a Internet, i per tant, si hi hagués problemes de connexió no es podria accedir aquests serveis i dades.

- Les dades sensibles no són locals, cosa que podria infringir una part de la Llei de protecció de dades, ja que exposen les dades a la possibilitat de robatori d'informació.

- La fiabilitat del servei recau sobre la tecnologia que utilitzen els proveïdors: moltes vegades el client no sap quins sistemes són els que donen servei i no sap si podrien ser més adequats pel mateix preu.

- La seguretat és també un punt clau: la informació ha de passar per diferents nodes per a arribar al destí i en cadascun d'aquests nodes hi poden haver vulnerabilitats.

3.3. Tipus de núvols

Hi ha diferents tipus de núvols (o models de desplegament) que cal detallar ja que cadascun té unes característiques diferents, i això provoca que s'ajustin més bé uns o altres depenent del tipus de client o empresa.

3.3.1. Núvols públics

El tipus de **núvol públic** consisteix en un proveïdor que ofereix serveis de computació per Internet de manera oberta a tothom que vulgui comprar els seus serveis.

Les **característiques** d'aquest tipus de núvols són les següents:

- **Disponibilitat**: a partir que es contracta el servei, en un breu lapse ja està disponible. A més, el proveïdor s'encarrega que un cop contractat estigui disponible durant el període contractat.

- **Pagament per ús**: permet contractar només aquells recursos que es necessiten, i així s'evita haver d'invertir en una infraestructura per a suplir aquesta necessitat.

- **Varietat de serveis**: permeten externalitzar totes les funcions bàsiques de l'usuari.

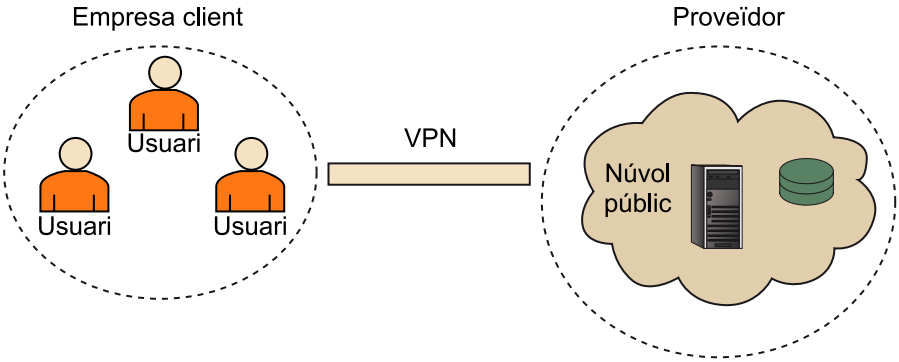
- **Seguretat**: el proveïdor és l'encarregat de complir els requisits de seguretat i protecció de dades.

- **Manteniment**: el proveïdor és l'encarregat de mantenir l'arquitectura necessària per a proporcionar el servei contractat.

- **Escalabilitat**: permet augmentar o reduir els serveis segons les necessitats del client al llarg del temps i amb un període d'implementació curt.

La figura 13 exemplifica la configuració de núvol públic.

Figura 13. Esquema de núvol públic genèric



3.3.2. Núvols privats

En el tipus de **núvol privat**, els recursos els gestiona el client mateix tot i que es pot trobar físicament a les seves pròpies instal·lacions o no.

Les **característiques** d'aquest tipus de núvols són les següents:

- **Reutilització**: permet aprofitar inversions de maquinari i personal fetes anteriorment.

- **Més temps d'implantació que el núvol públic**: preparar la infraestructura de núvol i redimensionar-la depenent de les necessitats requereix més temps que en el cas del núvol públic ja que l'usuari mateix s'ha d'encarregar de fer aquestes gestions.

- **Més especificitat**: el núvol es pot dissenyar ajustant-se més a les necessitats concretes de l'usuari.

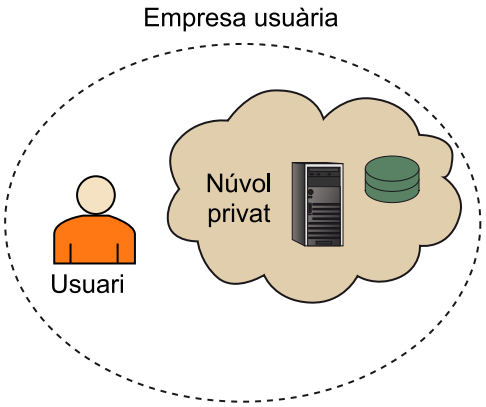
- **Gestió pròpia dels sistemes i les dades**: això implica que l'empresa s'ha d'encarregar del manteniment i la seguretat.

- **Amplada de banda**: com que es pot situar el núvol dins de l'organització mateixa s'eliminen les restriccions d'amplada de banda.

- **Seguretat**: si el núvol és només d'ús intern de l'usuari, es pot tenir aïllat d'Internet i així evitar possibles atacs externs. També s'obté més control sobre la gestió de la seguretat, ja que en el núvol públic es depèn de la gestió que en faci el proveïdor.

La figura següent exemplifica la configuració de núvol privat.

Figura 14. Esquema de núvol privat genèric



3.3.3. Núvols de comunitat

En el tipus de **núvol de comunitat**, el núvol és controlat i utilitzat per un grup d'organitzacions que tenen interessos comuns, com els requisits específics de seguretat o una funció comuna. Els membres de la comunitat comparteixen l'accés a les dades i aplicacions en el núvol.

Aquest tipus de núvols tenen unes **característiques** intermèdies entre el públic i el privat:

- **Més quantitat de recursos** que amb únic núvol privat, tot i que generalment menys que un núvol públic.
- **Especificitat**: el disseny del núvol pot ser més ajustat a unes necessitats concretes ja que les organitzacions que el formen tenen punts en comú.
- **Seguretat**: en termes de seguretat té l'avantatge respecte als núvols públics de ser d'accés més restringit, però tot i això és més obert que els privats.
- **Menys gestió de la infraestructura** i les dades que en un de privat però més que en un de públic.
- Requereix **inversió** en maquinari i programari.

3.3.4. Núvols híbrids

El tipus de **núvol híbrid** es caracteritza pel fet d'unir dos o més tipus de núvols diferents.

Aquests núvols continuen essent entitats úniques però interconnectades entre si que permeten gaudir dels avantatges dels diferents tipus de núvols:

- **Complexitat**: com que uneix diferents tipus de núvols la complexitat és més gran ja que cal integrar els diferents núvols amb les tecnologies que hi estan associades.

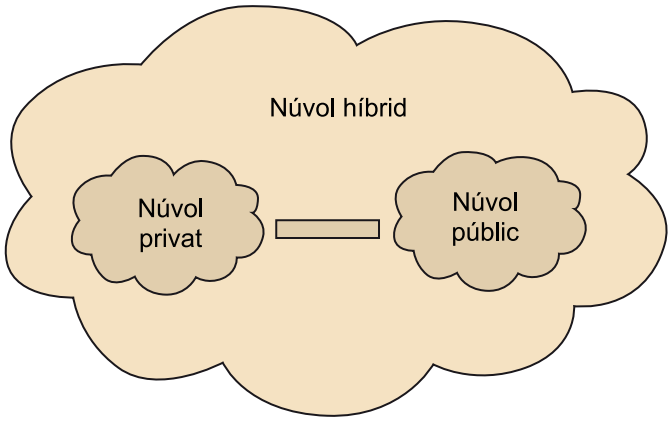
- **Flexibilitat i control**: permet més flexibilitat a l'hora de prestar serveis, a banda d'un control més gran d'aquests serveis i de les dades.

- Permet utilitzar cada tipus de núvol pels serveis que **s'adaptin més bé** a les característiques de cadascun.

Els núvols híbrids acostumen a intentar explotar primer els recursos propis (per evitar pagar per ús) i només quan l'usuari no té prou capacitat amb els recursos locals utilitza el núvol públic (per exemple, com a extensió o forma d'acceleració). Aquest tipus de tècnica s'acostuma a anomenar *cloud bursting*.

La figura següent exemplifica la configuració de núvol híbrid.

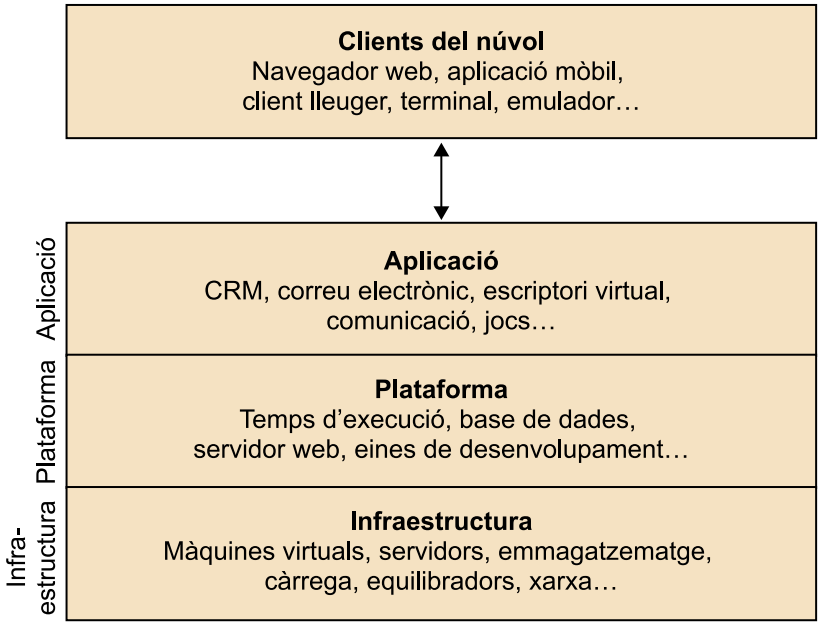
Figura 15. Esquema de núvol híbrid genèric



3.4. Models de servei

Tal com hem comentat abans hi ha tres models bàsics de provisió de serveis en núvol. Aquests models també s'acostumen a veure com una jerarquia per capes en què les capes superiors són més abstractes i properes a l'usuari (per exemple, el programari) i les capes inferiors, més properes al maquinari, tal com mostra la figura 16.

Figura 16. Model per capes de la computació en núvol



3.4.1. Infraestructura com a servei (IaaS)

En el model o capa d'infraestructura com a servei (IaaS[32]), el proveïdor ofereix un servei de computadores, màquines virtuals que posa a disposició del client.

[32]De l'anglès *infrastructure as a service*.

El servei consisteix, doncs, en tota la part de maquinari i sistema operatiu que el client ha de gestionar segons els seus interessos. El proveïdor només s'ha de preocupar de mantenir aquestes màquines en funcionament i connectades a Internet, i delega doncs tota la gestió de programari (sistema operatiu, bases de dades, servidors d'aplicacions) a l'usuari.

3.4.2. Plataforma com a servei (PaaS)

En el model o capa de plataforma com a servei (PaaS[33]), el servei va un pas més enllà del d'infraestructura com a servei i, en aquest model, el client rep la plataforma sencera (sistema operatiu, base de dades, servidor d'aplicacions, llenguatge de programació, entre d'altres) com a servei.

[33]De l'anglès *infrastructure as a service*.

En aquest tipus de servei el client només s'ha de preocupar de gestionar les aplicacions que vol fer córrer a la plataforma i no cal que gestioni conceptes relacionats amb la gestió ni el manteniment de la plataforma ni de la infraestructura.

3.4.3. Programari com a servei (SaaS)

El model o capa de programari com a servei (SaaS[34]) és el model preferiblement recomanat per a un client final i consisteix en l'adquisició del servei a més alt nivell, el qual correspon al nivell d'aplicació.

(34)De l'anglès *software as a service.*

En aquest servei el client només s'ha de preocupar de gestionar els paràmetres o configuracions de l'aplicació oferta pel servei, deixant de banda tant la màquina on resideix aquesta aplicació com el sistema operatiu que hi ha instal·lat o altres detalls que no són rellevants per a l'usuari, el qual no cal que en tingui cap coneixement més enllà de la utilització del programari.

Tal com es mostra en la figura 16, cadascun dels models o capes requereix el seu predecessor, ja que per a poder oferir aquest servei requereix l'anterior. Així doncs, si un proveïdor vol oferir un programari com a servei (SaaS), ha de disposar d'una plataforma com a servei (PaaS), o si no ha de contractar aquest servei amb un tercer. També amb la capa d'infraestructura com a servei, que és el model de més baix nivell. Aquest fet provoca que les vulnerabilitats i els forats de seguretat dels models de computació en núvol s'hagin de dividir segons el seu model i també s'hagin de gestionar segons aquest model, tant si l'encarregat de fer-ho és el proveïdor com el client. En aquest sentit, com a client, l'avantatge de disposar d'un servei de més alt nivell (SaaS) ofereix el desavantatge de no poder controlar la gestió de la seguretat d'aquest servei i a la inversa. Per exemple, si disposem d'un programari com a servei, perdem el control de la gestió del sistema operatiu, les actualitzacions d'aquest sistema, els requisits d'accés a les dades o la privacitat, mentre que si disposem d'un servei d'infraestructura serem nosaltres mateixos els qui haurem de gestionar tota la seguretat tant pel que fa al sistema operatiu com a les aplicacions i el programari.

També hi ha altres models de computació en núvol que neixen a partir de conjuncions dels tres models principals que hem vist. Els més rellevants són els següents:

- Xarxa com a servei (NaaS[35]).
- Emmagatzematge com a servei (STaaS[36]).
- Dades com a servei (DaaS[37]).
- Entorn de proves com a servei (TeaaS[38]).

(35)De l'anglès *network as a service.*

(36)De l'anglès *storage as a service.*

(37)De l'anglès *data as a service.*

(38)De l'anglès *test environment as a service.*

També en podríem dir d'altres de més especialitzats com ara de relacionats amb la seguretat o el monitoratge de múltiples aspectes.

Activitat

Busqueu informació sobre diferents proveïdors de servei en núvol i classifiqueu-los en funció del tipus de model de servei que ofereixen.

3.5. Casos d'ús

A partir del que hem vist anteriorment, en aquest subapartat hem fet una selecció dels que es consideren possibles casos d'ús que pugin provocar que un client o una empresa opti per un sistema en núvol:

1) Adaptar-se a les variacions estacionals de càrrega: les empreses, depenent del seu sector o sistema de funcionament, com el cas d'empreses que només treballen durant certs períodes de l'any o tenen pics importants de feina, tenen necessitats de tecnologies de la informació molt diverses i difícilment panificables. Aquest cas d'ús és molt important ja que utilitzar un núvol públic permet redimensionar la capacitat dels sistemes de tecnologies de la informació per tal d'adaptar-se a les variacions de carrega o a la necessitat de nous serveis. La capacitat del núvol per a aquest cas d'us és donada per la virtualització i la subcontractació[39]. La virtualització permet augmentar o disminuir els recursos de maquinari o serveis contractats per cada companyia de manera àgil, gracies al fet que proporciona independència del maquinari real.

[39] En anglès, *outsourcing*.

2) Canvi en el sistema de finançament: un altre cas d'ús popular de la computació en núvol és aprofitar-lo per a passar d'un sistema de finançament consistent a comprar actius fixos, per a aconseguir fer les tasques de tecnologies de la informació necessàries, a un sistema de pagar per serveis. D'aquesta manera s'obtenen diversos beneficis:

- Es redueix el risc. Tota inversió comporta un risc ja que la situació d'una companyia sempre pot variar a causa d'un element extern o intern i no es pot garantir que aquesta inversió es pugui amortitzar. Per això s'opta pel pagament de serveis ja que aquests serveis són costos a curt termini i que es poden anul·lar o modificar si en un futur es creu necessari.

- Aconseguir avantatge competitiu. En empreses petites o mitjanes, el núvol permet l'accés en les empreses a recursos que d'una altra manera haurien estat fora del seu abast a causa del gran cost que poden tenir.

- Com que el núvol encara està en un procés d'implantació les empreses opten per adherir-s'hi abans que la competència per tal de poder aportar un valor més gran que les altres. Aquest valor el poden aconseguir per exemple obtenint eines de gestió més bones que els permetin millorar el producte o servei, o aconseguint eines que els permetin un més bon contacte amb els clients.

3) Externalització de les tecnologies de la informació: l'externalització dels serveis de tecnologies de la informació ofereix una gran quantitat d'avantatges tot i que no està exempta d'alguns inconvenients, i per això hi ha algunes empreses que opten per solucions de núvol privat o híbrid. Els principals avantatges de l'externalització són els següents:

- Es redueix el personal necessari per al manteniment, encara que també pot ser extern en el sistema tradicional.

- No cal espai físic per als elements de maquinari, perquè l'usuari de núvol públic no necessita un espai físic per a mantenir els servidors ja que aquests servidors són virtualitzacions dels servidors del proveïdor.

- S'evita el manteniment tant de maquinari com de programari ja que el proveïdor s'encarrega de mantenir els serveis actualitzats i en funcionament.

4) Projectes amb un cicle de vida curt: per a les empreses que elaboren projectes, especialment si són curts i amb varietat de requeriments tant d'aplicacions com de capacitat de computació, adquirir tot el que fa falta per a tirar-los endavant els faria augmentar notablement el cost de desenvolupament d'aquests projectes ja que amb el breu temps de desenvolupament no es podria amortitzar una inversió en allò que fos necessari. Si una empresa fa aquest tipus de projectes li resultarà molt beneficiós un núvol públic, sia en conjunció amb un de privat o únicament el públic. D'aquesta manera poden calcular fàcilment el cost del lloguer dels serveis necessaris per al desenvolupament, durant el temps estimat, i afegir-los al pressupost. Així, s'aconsegueix un pressupost global més baix i més agilitat en els projectes, ja que es minimitza la implantació dels entorns necessaris.

5) Recuperació de desastres: després d'un desastre, per exemple natural, que hagi danyat el sistema informàtic d'una empresa, per a poder-ne reprendre el funcionament amb la màxima celeritat possible s'utilitza computació en núvol mentre es restableix el sistema de l'empresa. També s'utilitza a països en desenvolupament que tenen dificultats per a accedir a aquest tipus d'infraestructures o serveis de manera tradicional.

6) Necessitat de computació escalable massiva: les empreses que requereixen una capacitat de processament molt gran poden optar per utilitzar un servei IaaS sol·licitant la capacitat de computació necessària.

7) Consolidació d'infraestructura: necessitat d'una infraestructura de tecnologies de la informació sòlida capaç de resistir totes les necessitats de negoci actuals i amb flexibilitat per a adaptar-se a canvis futurs.

8) Accés universal: un altre element important del núvol és la possibilitat d'accedir als serveis del negoci des de diferents localitzacions i dispositius. Aquesta funció permet tant unificar més fàcilment les tecnologies de la informació de diferents seus en localitzacions separades –i aconseguir així que puguin accedir tots als mateixos recursos de tecnologies de la informació com si es tractés d'una única seu–, com la capacitat de controlar, modificar o utilitzar elements de les tecnologies de la informació des de fora de l'empresa amb diferents dispositius com mòbils o tauletes. Aquesta funció pot ser molt útil per a empreses que es dediquin a negocis en temps real com, per exemple, la compravenda d'accions. Poden fer operacions encara que no es trobin al seu lloc de treball, o tasques més simples com consultar el correu electrònic de l'empresa des de fora de la intranet.

9) Seguretat: per a la protecció de dades i les còpies de seguretat, les empreses es poden estimar més utilitzar un proveïdor que els garanteixi la seguretat i la protecció de les dades que implementar el seu mateix. Això es deu al fet que hi ha alguns proveïdors que deuen tenir més experiència en aquests temes que l'empresa mateixa. A més de la protecció de les dades contra atacs cal valorar que s'ofereix un sistema de replica de les dades per tal d'evitar pèrdues. Per a la mateixa empresa tenir un sistema de còpies de seguretat propi comportaria un cost important en maquinari i en la gestió d'aquest maquinari.

10) Ràpida implantació: la ràpida implantació es dóna principalment en els casos de núvol públic i permet a noves empreses iniciar el procés de negoci amb un temps més curt que si utilitzen un sistema de tecnologies de la informació tradicional, i també permet més flexibilitat a l'hora d'elaborar nous projectes, implementar nous serveis o passar de la infraestructura disponible al núvol.

11) Disminució del temps d'accés al mercat (*time to market*): en projectes com desenvolupaments informàtics, permet acurtar el temps entre les diferents fases de desenvolupament i test i facilitar així l'adquisició d'eines i la preparació d'entorns per a tirar endavant el procés, perquè d'una altra manera aquests canvis de fase podrien ser molt més costosos. Per acabar, en alguns casos també en pot facilitar la distribució, si es distribueix usant el núvol mateix.

12) Sistema de tecnologies de la informació compartit amb empreses amb interessos similars: els núvols compartits presentats en el subapartat anterior també comporten un motiu important per a decantar-se pel núvol, ja que hi ha conjunts d'empreses amb interessos comuns que poden compartir el núvol. Això permet crear un núvol més específic que un de públic ja que es pot invertir a crear-ne un que s'ajusti al màxim possible a les necessitats del conjunt. Com que es comparteix el núvol, això també facilita la compartició de dades i una capacitat més gran de cooperació.

13) Reducció d'emissions perilloses per al medi ambient: les empreses amb una gran quantitat de maquinari poden reduir de manera dràstica el seu consum energètic, i reduir així la contaminació que es provoca pel fet d'obtenir aquest tipus d'energia. Aquesta solució és fictícia si el proveïdor no utilitza energies renovables ja que redueix el consum de l'empresa però s'augmenta el del proveïdor.

3.6. Virtualització

Tal com hem comentat abans, la virtualització és una de les claus per a la computació en núvol ja que permet la multiplexació necessària per a aconseguir elasticitat i la il·lusió d'infinita capacitat. A continuació ens centrarem en els conceptes més bàsics de virtualització i després ens centrarem en les implicacions que tenen en la computació en núvol.

La **virtualització**, en un sentit ampli, consisteix en una capa d'abstracció de programari que s'afegeix entre el maquinari i el sistema operatiu que s'executa en el sistema. La capa resultant té les funcions següents:

1) Permet que s'executin simultàniament múltiples sistemes operatius en una única màquina física.

2) Permet particionar dinàmicament i aïllar la disposició dels recursos físics, com ara la memòria, els CPU, els discos i els dispositius d'entrada/sortida.

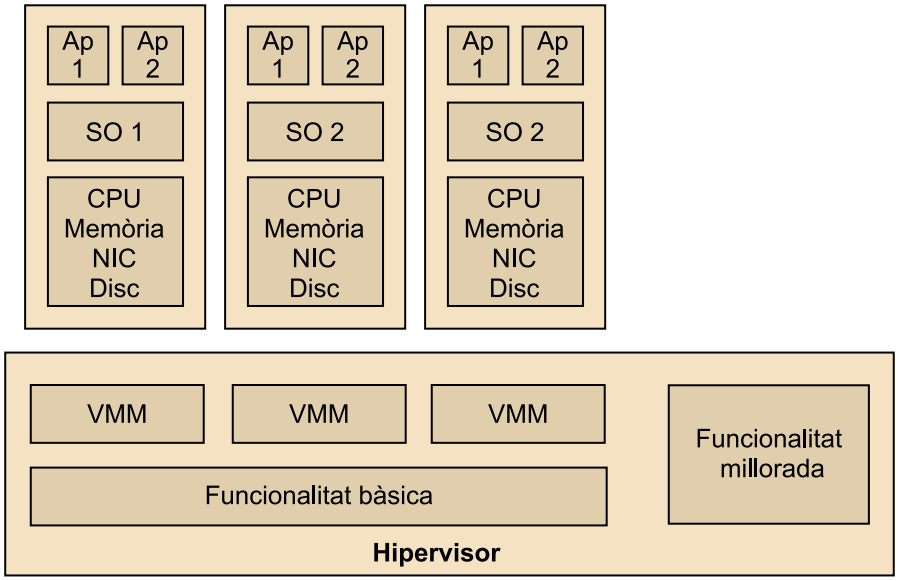
La virtualització és una tecnologia molt potent que ofereix grans beneficis, com ara permetre la consolidació de servidors, i proporciona una òptima utilització dels recursos, cosa que simplifica el desenvolupament de programari i l'elaboració de proves, i la millora de l'agilitat en els centres de dades.

Noteu que en aquest subapartat farem un èmfasi especial en la virtualització x86. La figura 17 mostra l'estructura bàsica d'un sistema virtualitzat. La capa de virtualització de base és el programari que organitza i gestiona diferents màquines virtuals en els monitors de màquines virtuals[40]. L'hipervisor és el component encarregat de la interacció directament amb el maquinari. La funcionalitat d'aquest component és diferent per a diferents arquitectures i implementacions. Cada VMM s'executa en l'hipervisor i té la seva pròpia abstracció de màquina virtual encarregada d'executar un sistema operatiu convidat[41]. Les seves funcions principals són la partició i compartició dels diferents recursos, com els dispositius de CPU, memòria, disc, i entrada/sortida de manera que tot el sistema és compartit per multiplexació entre diferents màquines virtuals.

(40)VMM, de l'anglès *virtual machine monitor*.

(41)En anglès, *guest operating system*.

Figura 17. Estructura bàsica d'un sistema virtualitzat



3.6.1. Tipus de virtualització

La virtualització de l'arquitectura x86 implica que el sistema de virtualització pugui treballar directament amb el maquinari, de manera que necessiten que el sistema tingui un control total dels recursos físics.

La figura 18 mostra els nivells de quatre nivells diferents de privilegis inherents a x86, coneguts com a *anell 0*, *1*, *2* i *3*, que denoten el nivell d'accés al maquinari de l'ordinador. L'anell 0 es fa servir principalment per a les aplicacions d'usuari, l'anell 3 per al sistema operatiu per a funcionalitats privilegiades que utilitzen directament els recursos del sistema, i els anells 1 i 2 són rarament utilitzats.

El tipus de virtualització també es pot classificar en les categories següents:

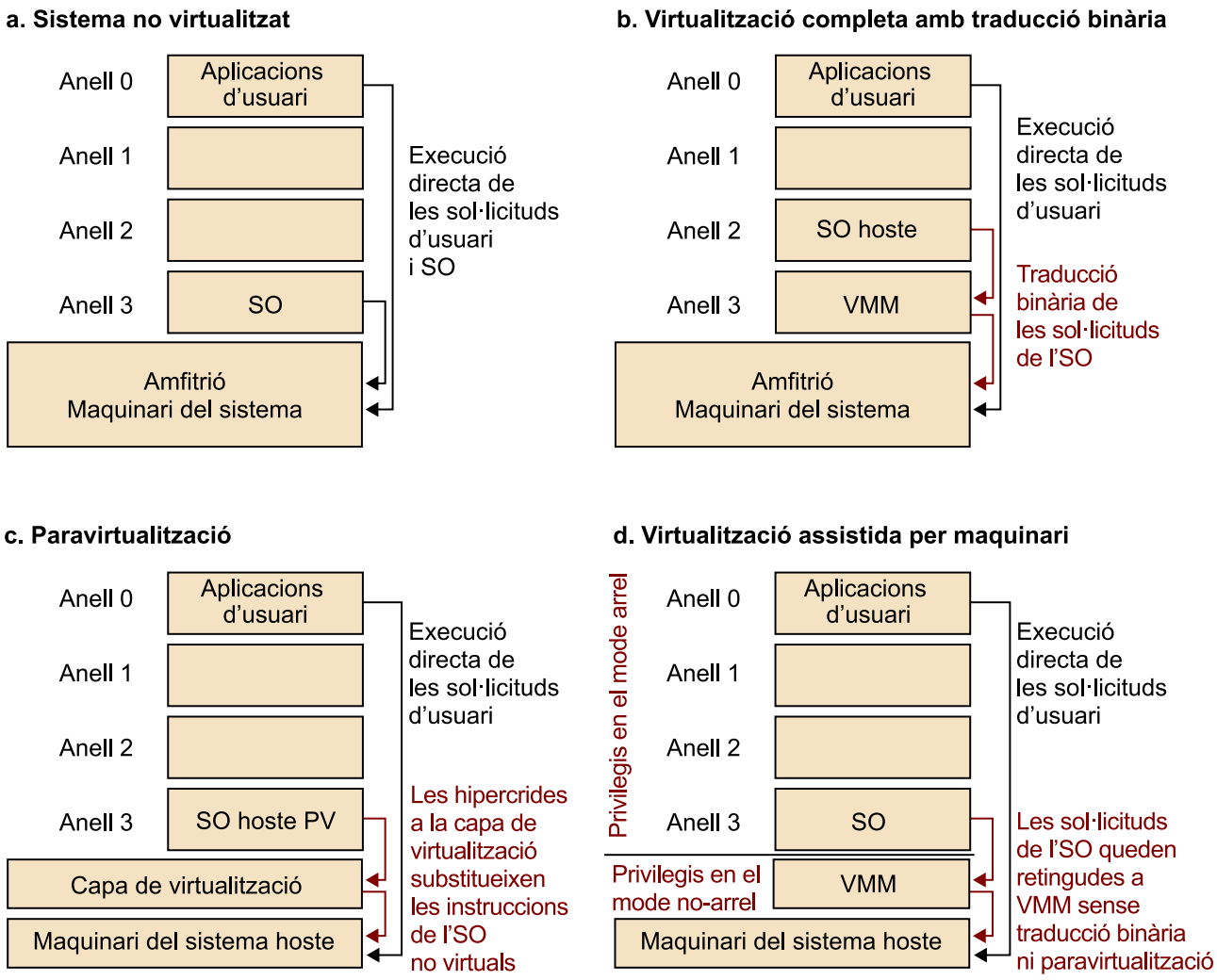
- Virtualització CPU.
- Virtualització de memòria i dispositiu.
- Virtualització d'entrada/sortida.

La virtualització de CPU es pot dividir en tres subtipus en funció dels mecanismes utilitzats per a gestionar les instruccions privilegiades i la flexibilitat per a assolir-ho. Aquests tres subtipus són els següents:

- Virtualització completa amb traducció binària.
- Paravirtualització.
- Virtualització assistida per maquinari.

La figura 18 mostra un sistema no virtualitzat i amb diferents tècniques per a aconseguir virtualització per mitjà dels anells de privilegis.

Figura 18



La virtualització completa amb traducció binària consisteix en una combinació de traducció binària (traducció de codi del nucli del sistema operatiu per a reemplaçar codi no virtualitzable amb un conjunt d'instruccions que tinguin el mateix efecte) amb execució directa (execució de codi d'usuari directament en el processador), de manera que cada VMM proporciona un conjunt de BIOS virtuals, dispositius i mòduls de gestió de memòria. Atès que l'hipervisor tradueix les instruccions del sistema operatiu en temps d'execució i deixa el codi d'usuari sense modificar, el sistema operatiu és independent del maquinari subjacent. Alguns dels beneficis obtinguts per aquesta tècnica inclouen un complet aïllament, d'alta seguretat, portabilitat i migració eficient de les màquines virtuals. Els productes de l'empresa VMWare i MS Virtual Server són uns exemples d'aquest tipus de virtualització.

La paravirtualització requereix modificacions en el nucli del sistema convidat, reemplaçant instruccions no virtualitzables per hipercrides que es comuniquen directament amb l'hipervisor per tal de fer operacions crítiques com,

per exemple, la gestió d'interrupcions. Tot i que no té prestacions de virtualització completa, la sobrecàrrega quant a rendiment és baixa i varia en funció de la càrrega de treball. Un exemple d'aquest tipus de virtualització és Xen.

La virtualització de memòria consisteix en la compartició i assignació dinàmica de memòria física a les màquines virtuals. De la mateixa manera que en la gestió de memòria proporcionada pels sistemes operatius moderns, hi ha un nivell addicional de memòria a causa de la virtualització de l'MMU. El VMM utilitza maquinari TLB per a mantenir les taules de pàgines per a una cerca directa durant el mapatge del sistema operatiu convidat a la memòria física.

En la virtualització de dispositiu i d'entrada/sortida gestiona l'encaminament de sol·licituds d'entrada/sortida entre els dispositius virtuals i el maquinari físic compartit que permet una gran quantitat de prestacions amb una administració senzilla. Els dispositius virtuals poden interactuar entre si sense problemes i sense afectar els recursos físics i es poden migrar o manipular de manera espontània. L'hipervisor estandarditza tots els dispositius virtuals que permetin la portabilitat entre plataformes a causa d'una configuració compatible de maquinari virtual en qualsevol maquinari físic.

La virtualització ofereix diversos **avantatges** respecte als sistemes tradicionals, entre els quals trobem els següents:

- **Facilitat de depuració**. Els desenvolupadors poden provar nous sistemes operatius en màquines estables i també poden depurar el seu codi amb més eficàcia a causa de l'aïllament dels diferents sistemes. La recuperació també és més ràpida mitjançant l'ús de versions guardades dels clients.

- **Balanceig de la càrrega i reubicació**. Les màquines virtuals es poden migrar entre equips per a equilibrar la càrrega de treball i també permeten agregar diferents recursos per millorar-ne l'eficiència.

- **Consolidació de servidors**. Una única màquina física pot executar moltes màquines virtuals. Cada màquina virtual té el seu propi accés a l'arrel i pot triar els tipus d'aplicacions o serveis que s'executin sense dependre d'altres usuaris. Això també fa que es maximitzi la utilització dels sistemes multiprocessadors. L'arrencada d'un sistema operatiu resulta tan simple com iniciar una aplicació.

- **Menys cost de propietat**. La utilització òptima dels recursos físics redueix el cost total de propietat de les empreses i també redueix els costos de formació dels treballadors a causa de la reconfiguració mínima dels sistemes. Els usuaris poden executar productes heretats en un entorn segur en noves arquitectures i estalviar en els costos de refinament de l'aplicació.

- **Menys consum d'energia i infraestructura de refrigeració**. La utilització eficient de recursos es tradueix en un ús òptim de l'energia i la reducció

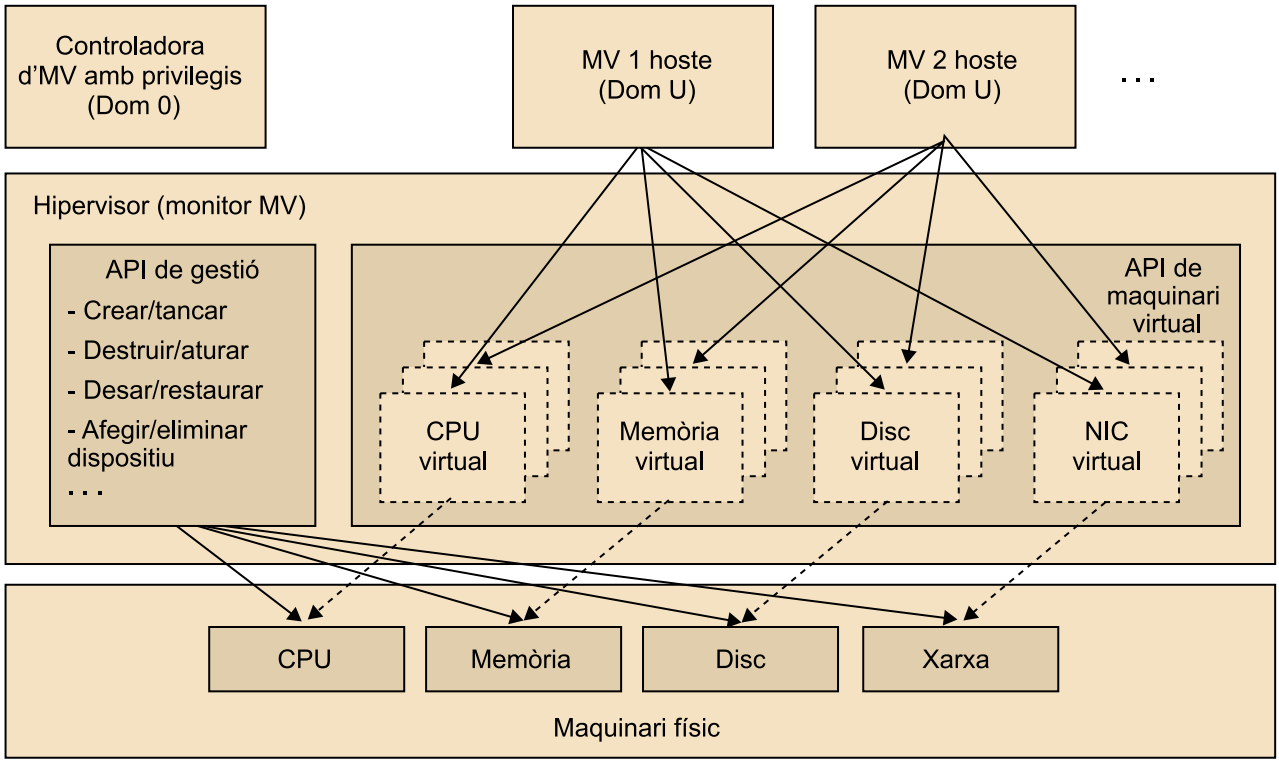
dels seus costos. Disposar d'una infraestructura més petita ens permet una eficaç refrigeració dels centres de dades amb menys sistemes d'aire condicionat, cosa que redueix els costos energètics.

En aquest subapartat utilitzarem Xen com a cas d'ús de sistema de virtualització, un dels més utilitzats (incloent-hi núvols públics com Amazon EC2).

L'arquitectura de Xen consta de tres components principals per sobre del maquinari, tal com mostra la figura 19. Aquesta arquitectura té tres components bàsics:

- Hipervisor Xen / monitor de màquines virtuals.
- Domini privilegiat / conductor (Dom 0 en la figura).
- Dominis d'usuari / convidats (Dom U en la figura).

Figura 19. Esquema general de l'arquitectura Xen



L'hipervisor o monitor de màquina virtual (VMM) de Xen és el component que té relació directa amb el maquinari físic i que proporciona una interfície virtual als dominis que Xen allotja. Les **funcions bàsiques** que té són les següents:

- Cada domini rep una porció específica dels recursos de la màquina física. Aquesta distribució de recursos pot ser arbitrària, equitativa o seguir alguna directiva d'usuari. El VMM pot optar per restringir l'accés a algun dispositiu físic d'un domini convidat o fins i tot pot crear un dispositiu virtual que no existeix.

- El VMM ofereix dispositius virtuals als dominis. La fabricació d'un dispositiu té una importància secundària, ja que es tracta d'un dispositiu de tipus general, un dispositiu de xarxa o un dispositiu de blocs en cas de dispositius d'emmagatzematge.

- El VMM és capaç de modificar parts de l'arquitectura d'acollida que són difícils de virtualitzar. Això requereix canvis en el sistema operatiu convidat, però no programari d'usuari.

Per a dur a terme aquestes funcions, el VMM ocupa una posició privilegiada en el sistema a partir d'una arquitectura basada en capes que hem discutit abans quan parlàvem de paravirtualització.

El domini privilegiat o conductor (domini 0) és un sistema operatiu especial privilegiat que fa tasques administratives per a l'hipervisor. Llançat en l'arrencada, és l'eina principal per a manipular i migrar dominis convidats entre màquines físiques. Dom 0 crea conductors ideals per a dispositius de xarxa i de blocs de tots els convidats i es comunica amb els clients per mitjà de transport asíncron per memòria compartida.

El Dom 0 també executa un controlador en segon pla que s'encarrega de proporcionar a cada convidat una interfície genèrica per tal de crear la il·lusió que el dispositiu està dedicat a cada màquina virtual. En Dom 0 hi ha dues eines per a la gestió de les màquines virtuals:

1) xend (*xen daemon*): es tracta d'un procés crític de Xen que s'executa com a arrel (*root*) en Dom 0. Proporciona una interfície anomenada *xm* (*xen management*) als usuaris finals amb la finalitat de crear, tancar o manipular dominis i configurar-ne els recursos. Tant xend com xm són *scripts* de Python.

2) xenstored (*xen store daemon*): Xen proporciona una manera de compartir la informació de configuració entre diversos convidats per al manteniment d'una base de dades comuna que es diu *XenStore*. S'utilitza sobretot per a controlar dispositius en dominis convidats, dur a terme operacions atòmiques com la lectura o escriptura de les claus i manipular l'estat de configuració entre els conductors.

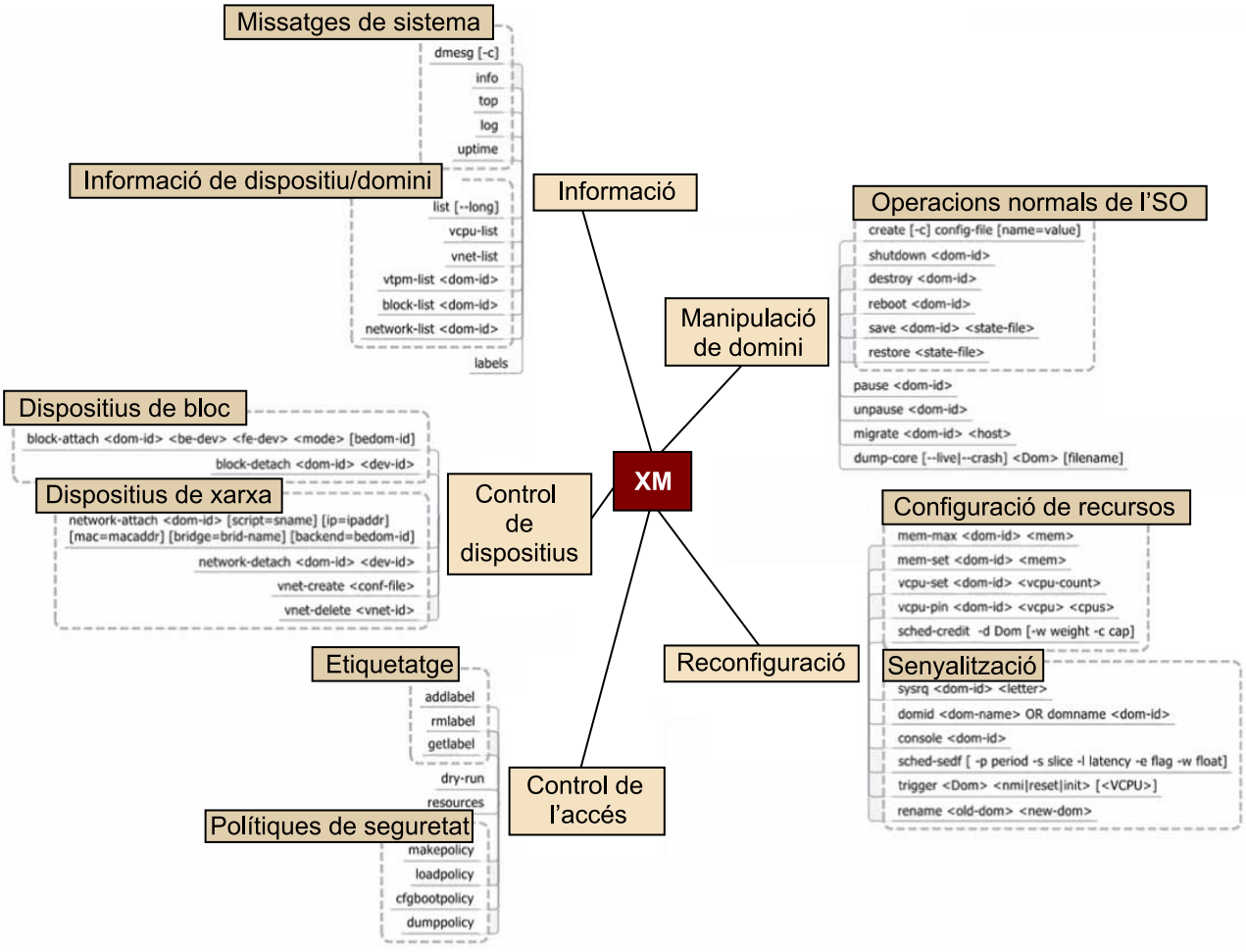
Els sistemes operatius modificats que s'executen sobre un VMM gestionat per Dom 0 es denominen *dominis convidats* o *d'usuari* (Dom U). Hi ha diverses aplicacions d'usuari que s'executen en Dom U, com, per exemple, aquelles que interactuen amb els controladors de dispositius. A continuació veurem breument algunes eines de Xen que tenen una gran quantitat de funcionalitats dirigides a controlar o monitorar el comportament d'aquestes màquines virtuals convidades i del VMM en general.

L'ordre xm s'utilitza per a l'administració dels dominis convidats en Xen. Les principals **funcions** d'aquesta ordre (agrupades per categories en la figura 20) són les següents:

- Gestió dels diferents dispositius.
- Manipulació de les màquines virtuals convidades i reconfiguració de diferents recursos com CPU o memòria.
- Disseny de polítiques.

La informació del sistema també es pot consultar per motius de depuració o de control.

Figura 20. Funcions de l'ordre xm de Xen



Xen proporciona una eina anomenada *XenTop* semblant al *top* característic dels sistemes Linux que es pot executar com a programa interactiu en totes les màquines virtuals en execució i utilitza un format de taula amb estadístiques de consum de recursos. És una eina fàcil d'usar per a comprendre ràpidament l'estat del sistema i per a observar l'efecte de certes accions. La llista de la informació proporcionada és dinàmica i pot incloure el percentatge d'ús de CPU i memòria, trànsit de xarxa, etc.

També hi ha l'eina XenMon, que s'encarrega de monitorar la qualitat de servei proporcionada i en general del rendiment. Basat en aquest monitoratge també permet fer certes accions per tal de complir els objectius de les polítiques establertes.

3.7. Computació en núvol per a altres prestacions

Tal com hem vist abans, la computació en núvol està revolucionant el món de l'empresa de la mateixa manera que la Internet ho va fer no fa tant de temps. La computació en núvol està canviant aspectes fonamentals de la manera com les empreses gestionen la infraestructura de les tecnologies de la informació, tant internament com externament, i proporciona accés per encàrrec als serveis de computació amb la capacitat d'escalar els recursos quan és necessari mitjançant l'externalització d'infraestructura.

Al mateix temps que la computació en núvol està redefinint les tecnologies de la informació, la creixent escala de computació i volums de dades està canviant la manera en què es fa ciència i recerca en enginyeria amb nous paradigmes i pràctiques que es basen essencialment en les dades i la col·laboració per a afrontar grans reptes a escala internacional. S'espera que els serveis en núvol es podran unir als sistemes clúster tradicionals d'altes prestacions i sistemes de còmput en graella per tal de fer descobriments científics clau.

Els núvols poden permetre l'externalització o subcontractació de molts aspectes de la recerca i de l'educació que no són realment útils, com ara la implementació, configuració i administració de la infraestructura, i així permetre als científics centrar-se en la ciència. Els serveis en núvol també poden millorar la productivitat, facilitar l'intercanvi de resultats de recerca, i permetre la reproductibilitat dels càlculs que hi estan associats. D'altra banda, més similar a l'entorn empresarial, els núvols poden democratitzar l'accés als recursos computacionals i de dades (mitjançant l'accés als investigadors que no tenen la infraestructura local adequada), cosa que ha demostrat un impacte significatiu en la productivitat de la recerca (IBM, 2009). De fet, una recent enquesta d'usuaris del Departament d'Energia dels Estats Units duta a terme per l'equip Magellan va trobar que les motivacions principals per al moviment dels usuaris cap al núvol era la facilitat d'accés als recursos informàtics (esmentat pel 79%), la capacitat de controlar els entorns de programari (59%), i la capacitat de compartir la configuració de programari i els experiments amb els companys (52%).

No obstant això, també és important mirar més enllà dels beneficis de l'externalització i comprendre les possibles maneres de formular les aplicacions i d'utilitzar els recursos en un entorn híbrid on es poden combinar sistemes d'altes prestacions tradicionals, sistemes de computació en graella i sistemes de computació en núvol.

La computació en núvol s'està sumant als sistemes d'altes prestacions de tipus clúster i en graella com a plataformes viables per a aplicacions de ciència i enginyeria. Així doncs, és especialment important descobrir noves formes d'ús de tota aquesta infraestructura per tal de donar suport a aplicacions de la ciència i enginyeria. Per a dur-ho a terme és important entendre bé les aplicacions i els requeriments d'aquestes aplicacions. Per exemple, cal tenir clar si es tracta d'aplicacions d'altes prestacions de computació (HPC), de cabal o *throughput* (HTC) o massiva de tasques (MTC) (Raicu i altres, 2008). Les aplicacions HPC estan fortament acoblades amb grans quantitats de comunicació interprocessador, i típicament requereixen grans quantitats de potència de càlcul durant períodes curts. D'altra banda, les aplicacions d'HTC són generalment dèbilment acoblades, i la comunicació entre processadors és limitada o inexistent. Les aplicacions HTC també requereixen grans quantitats de computació, però per temps molt més llargs (mesos o anys, en lloc d'hores o dies). Finalment, les aplicacions d'MTC són un híbrid de les dues classes anteriors. Les aplicacions MTC poden consistir en tasques d'acoblament flexible, en què cadascuna d'aquestes tasques és una aplicació ben acoblada que s'executa durant un període curt (és a dir, segons o minuts).

Els serveis de computació en núvol poden funcionar amb aplicacions científiques de múltiples formes. Pot proporcionar una plataforma per a les aplicacions, per exemple, quan la infraestructura local no està disponible. També pot servir per a complementar plataformes existents per a proporcionar capacitat addicional o capacitats complementàries per a satisfer les necessitats heterogènies i dinàmiques. Per exemple, els núvols poden servir com a acceleradors, o proporcionar més resiliència pel fet de poder disposar de recursos addicionals en cas que es produeixi un error. De fet, la computació en núvol no solament pot ajudar els científics a resoldre els problemes d'avui amb més eficàcia, sinó que també pot permetre explorar noves maneres de formular els problemes mitjançant les abstraccions de la computació en núvol com són l'elasticitat o el model de pagament per ús.

Cal tenir en compte que ens centrem en núvols de computació però la mateixa discussió també es pot aplicar als núvols de dades; la idea és poder funcionar amb dades científiques i d'enginyeria i proporcionar l'anàlisi de dades com a servei.

Entre els possibles models que combinen la computació en núvol amb els conceptes més tradicionals de la computació d'altres prestacions, hi podem trobar els següents:

1) HPC en núvol, en què els investigadors poden externalitzar aplicacions completes al núvol o plataformes de núvol privades. Els sistemes en núvol actuals poden resultar eficaços per a certes classes d'aplicacions, com, per exemple, aplicacions HTC. Un recent informe tècnic (Fox i Gannon, 2012) ha estudiat de manera exhaustiva executar aplicacions HPC en el núvol. Segons aquest estudi, la computació en núvol només funciona eficaçment per a certs

tipus d'aplicacions HPC. Un exemple d'això són les aplicacions *embarrassingly parallel*, que analitzen dades independents o generen simulacions independents que integren dades de sensors distribuïts, o d'anàlisi de dades que poden utilitzar models de tipus MapReduce. Uns altres treballs de recerca (Fox, 2011; Iosup i altres, 2011) mostren que les diferents variants de càlculs de MapReduce i MapReduce iteratiu funcionen bé en núvol. En general, les aplicacions HPC amb sincronització mínima i requisits de comunicacions mínimes, amb poca entrada/sortida i d'escales modestes són adequades per a les plataformes de núvol actuals i, per tant, poden ser externalitzades al núvol amb èxit. El cost i el rendiment relatiu pot esdevenir un problema. Per exemple, l'informe del projecte Magellan (Yelick i altres, 2011) cita que els serveis en núvol van resultar de 7 a 13 vegades més cars.

2) HPC més núvol se centra en l'exploració d'escenaris en què els núvols poden complementar recursos HPC / en graella amb els serveis en núvol per donar suport a aplicacions de ciència i d'enginyeria, per exemple, per fer de suport als requeriments heterogenis o a pics inesperats en la demanda. Un núvol híbrid més infraestructura HPC tradicional també pot permetre, potencialment, noves formulacions per a aplicacions que utilitzen núvols com acceleradors, per a resiliència o per a poder balancejar el compromisos entre cost, consum elèctric i rendiment.

3) HPC com un núvol, centrat a exposar recursos HPC / en graella utilitzant les abstraccions de núvol, amb l'objectiu de combinar la flexibilitat dels models del núvol amb el rendiment dels sistemes HPC. A causa de les limitacions dels núvols a l'hora de funcionar amb aplicacions HPC directament, els proveïdors del núvol es van adonar de la necessitat de proporcionar solucions en núvol que es construeixen específicament per a aplicacions HPC (és a dir, el maquinari amb processadors i interconnexions més ràpids). Hi ha alguns proveïdors que han proporcionat fins i tot maquinari no virtualitzat per tal de proporcionar el rendiment que necessiten aquestes aplicacions. Això es coneix com a *HPC com un núvol* (és a dir, executar aplicacions HPC en els recursos que s'exposen com a recursos per encàrrec utilitzant abstraccions del núvol, per tal d'aprofitar el model de núvol sense sacrificar el rendiment HPC que requereixen les aplicacions científiques). Hi ha dos enfocaments principals que s'utilitzen per a proporcionar HPC com un núvol. El primer enfocament utilitza grans sistemes HPC que es poden aprovisionar en forma de núvols. El segon enfocament utilitza petits clústers HPC que es poden connectar entre si per a formar un gran núvol. Aquests clústers HPC poden ser virtualitzats o no virtualitzats per a oferir un rendiment més bo.

Bibliografia

Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A. D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; Zaharia, M. (2010). "A view of cloud computing". *Commun. ACM* (vol. 4, núm. 53, pàg. 50-58).

Foster, I.; Yong Zhao; Raicu, I.; Lu, S. (2008). "Cloud Computing and Grid Computing 360-Degree Compared. Grid Computing Environments Workshop, 2008". *GCE 08* (pàg. 1-10, 12-16).

Foster, I.; Kesselman, C. (1999). *The Grid: Blueprint for a New Computing Infrastructure*. Burlington, Massachusetts: Morgan Kaufmann.

Foster, I.; Kesselman, C.; Tuecke, S. (2001). "The Anatomy of the Grid, Enabling Scalable Virtual Organizations". *International Journal of High Performance Computing Applications* (vol. 15, núm. 3).

Fox, G. (2011). "Data Intensive Applications on Clouds". A: *Keynote at The Second International Workshop on Data Intensive Computing in the Clouds (DataCloud-SC11) at SC11 November 14*.

Fox, G.; Gannon, D. (2012). "Cloud Programming Paradigms for Technical Computing Applications". Informe tècnic.

Goyal, A.; Dadizadeh, S. (2009). "A Survey on Cloud Computing". Colúmbia Britànica: The University of British Columbia.

Hwang, K.; Dongarra, J.; Fox, G. (2011). *Distributed and Cloud Computing*. Burlington, Massachusetts: Morgan Kaufmann.

IBM (2009). "IBM Research Doubles Its Productivity with Cloud Computing".*Case Study QuickView*.

Iosup, A. i altres (2011). "Performance analysis of Cloud computing services for many-tasks scientific computing". *IEEE TPDS*. Los Alamitos, Califòrnia.

Iverson, W. (2004). *Real World Web Services* [en línia]. Sebastopol, Califòrnia: O'Reilly.

Mell, P.; Grance, T. (2011). *The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology*. Estats Units d'Amèrica: US Department of Commerce.

Monson-Haefe, R. (2004). *J2EE Web services*. Reading, Massachusetts: Addison-Wesley.

Raicu, I.; Zhang, Z.; Wilde, M.; Foster, I.; Beckman, P.; Iskra, K.; Clifford, B. (2008). "Towards Loosely-Coupled Programming on Petascale Systems". *IEEE/ACM Supercomputing*.

Yelick, K. i altres (2011). *The Magellan Report on Cloud Computing for Science*. Estats Units d'Amèrica: US Department of Energy Office of Advanced Scientific Computing Research (ASCR).