Sisteme integrate de servicii distribuite. Studii de caz

Radu SION
http://sunsite.pub.ro/radu

În cadrul acestui articol ne propunem analiza unor tendinte de actualitate în domeniul sistemelor de operare distribuite, anume convergenta si integrarea unor notiuni ca sistem de operare, sistem distribuit, prelucrare paralela, servicii. Aparitia unor capacitati de calcul cu un raport putere/pret extrem de ridicat, a unor echipamente de comunicatie fiabile, cu rate de transfer din ce în ce mai mari, precum si a unor solutii software integrate ce asigura portabilitatea aplicatiilor precum si transportabilitatea codului prin intermediul unei retele de calculatoare, a constituit masa critica necesara aparitiei unei noi etape în evolutia sistemelor de calcul în ansamblu si a celor de operare în special. Pentru informatii suplimentare se pot consulta urmatoarele resurse:
***Jet Virtual Medium** (http://jvm.sunsite.pub.ro), **Talking Objects** (http://talking.sunsite.pub.ro), **IQ** (http://iq.sunsite.pub.ro)*
***Cuvinte cheie:** distributed systems, servicing, distribuited computations, virtual comunities, computer networks.*

Introducere

Pentru a putea fi utilizate eficient, resursele de comunicatie existente au fost supuse de-a lungul timpului unui îndelungat proces de rafinare si integrare, proces ce a dus în final la aparitia unei importante tendinte de convergenta catre solutii stabile, multi-functionale. Exemple simple în acest sens ar fi chiar cele legate de evolutia, în ultimii 10-15 ani, a protocoalelor de comunicatie. Astfel, daca în anii de început ai Internetului existau o mare varietate de protocoale ce concurau practic pentru anumite functionalitati de baza, (de exemplu hipertext, email) anii '90 au adus conceptul de World Wide Web implementat prin noi instrumente gen web-browser etc. Acest nou mediu a preluat extrem de rapid conducerea în topul preferintelor utilizatorilor, pentru ca apoi sa integreze putin cîte putin, majoritatea celorlalte protocoale, într-o interfata unica.

Disparitia multor protocoale redundante s-a facut simultan cu aparitia altora, rezolvând necesitati noi ale comunitatilor de utilizatori. De data aceasta însa, proiectarea noilor protocoale s-a facut clar în ideea integrarii de la început cu tehnologiile existente. Se porneste de la nivelul *proiectarii de retea* si se construiesc noi modele si protocoale, integrate prin intermediul unor sistem de tip distribuit. Un asemenea model implica o comunitate virtuala bazata pe retele, "caramida" comunicatiei fiind obiectul, în locul simplului stream de biti, aparatele fiind reprezentate de catre entitati virtuale, oferind într-o maniera transparenta acces la toate facilitatile intrinseci ale hardware-ului, serviciile find oferite utilizînd scheme de servicing distribuit, implicînd enduser-ul la un nivel minim în localizarea si prelucrarea intermediara a resurselor.

În cele ce urmeaza încercam trecerea în revista a unor implementari existente, ce converg catre aceasta viziune, atât prin facilitatile oferite cât si prin aplicatiile ce le utilizeaza.

Lucrarea este structurata în felul urmator: capitolul 2 prezinta principalele concepte cheie ce apar în cadrul majoritatii implementarilor existente, exemplificând apoi cu studii de caz individuale, pentru fiecare dintre sistemele prezentate. Sunt prezentate notiuni precum *lookup*, *repository*, *service* etc. Capitolul 3 încearca prezentarea unor

elemente specifice fiecarui sistem în parte. Sunt prezentate aici puncte cheie din sisteme precum CORBA, Jini, Talking Objects etc. În capitolul 4 sunt discutate elemente înca controversate precum scalabilitatea sau portabilitatea unor asemenea sisteme.

Concepte cheie – studii de caz

În cele ce urmeaza sunt prezentate principalele concepte cheie ce apar în cadrul majoritatii implementarilor existente, urmând apoi exemplificari cu studii de caz individuale.

Sunt studiate urmatoarele sisteme: ***CORBA*** (Common Object Borker Request Architecture) dezvoltat de OMG (Object Management Group) pentru un întreg grup de firme reunite într-un effort anti-Microsoft, ***Jini*** de la Sun Microsystems, ***KnittingFactory***, o realizare academica de la New York University, ***Calypso*** si ***Charlotte***, doua sisteme realizate în mare parte de aceeasi echipa cu cea care a realizat KnittingFactory, ***BOND***, un sistem realizat la Purdue University, ***InfoSpheres***, de la California Institute of Technology, precum si ***Talking Objects***, un sistem original proiectat de noi.

Sisteme distribuite

Pentru definitii unanim acceptate ale notiunii de sistem distribuit putem consulta diverse referinte precum [7] si [8]. În esenta însa dorim de asemenea delimitarea anumitor particularitati ale notiunii, particularitati relevante în cazul studiului de fata. Pentru aceasta vom încerca o definire sumara a notiunii.

De-a lungul timpului s-a conturat un set de clasificari definind modalitatea de procesare a informatiei într-un anumit sistem. Una dintre acestea avea în principal în vedere, nivelul de interconectare fizica si logica a elementelor de procesare individuale, intr-un sistem multiprocesor. Astfel gasim la una dintre extreme, asa numitele sisteme *tightly-coupled* (cu legaturi strânse) iar la cealalta extrema, sistemele *loosely coupled* (cu legaturi slabe). În mod curent, sistemele aflate în vecinatatea primei extreme sunt cunoscute sub denumirea de *sisteme de procesare paralela* [7], pe când cele din vecinatatea celei de-a doua, sunt numite *sisteme de calcul distribuit* [8].

În prezentul studiu ne vom delimita putin de cele afirmate mai sus. Vom întelege prin sistem distribuit o *constructie logica* incluzând noduri active de procesare interconectate prin legaturi cu caracteristici diferite, nodurile având un oarecare grad de independenta din punctul de vedere al efectuarii unor operatii de calcul. De asemenea vom presupune localizarea fizica a resurselor de calcul ca fiind de o raspândire mult mai mare decat în cazul unei simple structuri cluster locale. O ultima caracteristica necesara ar mai fi existenta unei coerente temporale în prelucrarile din interiorul sistemului.

Este de remarcat faptul ca, odata cu aparitia unor retele rapide (ex. de tipul ATM), este tot mai greu de facut distinctie între procesarea paralela masiva locala (HPC) si procesarea distribuita. În fapt multe dintre sistemele HPC curente sunt bazate pe solutii distribuite, într-un micro-univers local.

Este relevanta existenta unor caracteristici extrem de importante, comune si totodata definitorii pentru majoritatea implementarilor cunoscute de sisteme distribuite. De-oarece toate sistemele prezentate sunt în esenta bazate pe o structura de calcul si comunicatie distribuita, vom încerca în continuare analizarea unor caracteristici de baza în contextul fiecaruia dintre ele.

Comunicatie distribuita

Notiunea de ***comunicatie distribuita*** se refera în principal la posibilitatea transferului inter-entitati a unor unitati informationale, în majoritatea cazurilor încapsulate într-un obiect, identificat printr-un identificator unic. Vom considera atât cazul utilizarii unor tehnici de *message-passing* sau ***RPC*** *(Remote Procedure Call)* cât si cazul

utilizarii unor facilitati partajate, cum ar fi *memoria distribuita* partajata, desi acestea pot fi implementate unele prin intermediul celorlalte.

Unul dintre sistemele des întâlnite este ***CORBA*** (Common Object Request Broker Architecture). În cadrul ***CORBA*** comunicatia distribuita este implementata prin utilizarea unui asa numit ***ORB*** (Object Request Broker) care la rândul lui poate fi implementat deasupra altor solutii de comunicare distribuita. ***ORB***-ul va asigura transparenta accesului oricarui client ***CORBA*** la obiectul serviciu dorit.

Tehnologia ***Jini***, fiind în totalitatea sa bazata pe Java, foloseste din plin facilitatile oferite de aceasta, prin ***JavaSpaces*** [12] si ***JavaRMI*** [14] (***JavaRMI*** reprezinta corespondentul RPC în cadrul tehnologiei Java). La un nivel inferior, comunicatia se bazeaza în esenta pe serializarea obiectelor transmise în retea. O caracteristica interesanta a ***Jini***, ce va fi discutata în continuare (a se vedea 2.2.1), este notiunea de *federatie*.

KnittingFactory, fiind în principal un sistem ce implementeaza o paradigma asemanatoare cu *replicated workers*, va utiliza în principal structura applet din Java pentru efectuarea prelucrarilor efective. Pentru comunicatie va folosi atât structura web HTTP existenta, în procesul de înregistrare a noilor unitati de calcul cât si în cel de cautare a celor existenti la un anumit moment dat, iar în cadrul comunicatiei inter-worker va folosi o aplicatie server speciala, ce rezida pe calculatorul de unde au fost încarcati cei doi *workeri*. Exista însa o serie de probleme nerezolvate în acest sistem.

Sistemele ***Calypso*** si ***Charlotte*** se ocupa de asemenea cu prelucrari paralele distribuite, definite însa într-un cadru mai restrâns decât cel propus de ***Jini*** spre exemplu. Astfel ambele sisteme se bazeaza, în implementarea comunicatiei distribuite pe existenta unor procese daemon (Unix respectiv Java) ce asigura comunicatia prin intermediul TCP/IP sau UDP.

Arhitectura sistemului ***BOND*** faciliteaza practic utilizarea oricarui middle-ware de comunicatie distribuita. În versiunea curenta, ***BOND*** utilizeaza serviciile de comunicatie existente în Infospheres, prezentate în continuare. În esenta, partea de comunicatie utilizata de BOND este o implementare ce ofera serializare de obiecte.

Infospheres, a carui versiune initiala a precedat aparitia lui ***Jini*** sau a ***JavaRMI***, utilizeaza (în forma initiala) majoritatea resurselor existente în cadrul unei structuri Internet-Java-HTTP. Astfel serializarea de obiecte, accesul direct TCP/IP precum si utilizarea structurii Web se regasesc în Infospheres. Un element interesant în Infospheres sunt notiunile de *inqueue* si *outqueue* (cozi de mesaje) aferente fiecarui obiect în parte (Acestea sunt referite în documentatia ***InfoSpheres*** [10] si ca *inbox* / *outbox*, cu semnificatia de cutii postale). Ele sunt utilizate în transmiterea/receptionarea mesajelor obiectului respectiv.

Talking Objects, o specificatie de nivel înalt, conceputa pentru a putea utiliza virtual orice sistem de comunicatie distribuita, introduce un set de noi elemente ce necesita rezolvare (de exemplu una dintre acestea ar fi necesitatea existentei unor identificatori globali unici de tip URL, URN etc. pentru orice entitate/obiect adresabila din cadrul unei formulari în limbaj natural a unui serviciu). În prezent ***Talking Objects*** utilizeaza serviciile oferite de ***Jet Virtual Medium*** [6], un sistem complex de comunicatie distribuita, bazat pe Java si serializare de obiecte. Sistemul de comunicatie este accesibil la ***http://jvm.sunsite.pub.ro***.

Securitate

Securitatea este un element extrem de important în toate sistemele curente de interes. Dezvoltarea unei întregi industrii în domeniu, finalizând peste o duzina de importante standarde si protocoale de

comunicatie sigura (secure), demonstreaza foarte clar acest lucru.

Din pacate, în cadrul multor sisteme distribuite, securizarea nu a fost considerata un element crucial în proiectarea initiala, poate si datorita caracteristicii de prototip a multora dintre ele. Exista însa si sisteme (în principal cele de data recenta) ce au acordat atentia cuvenita acestui aspect.

CORBA implementeaza o schema de securitate extrem de populara în prezent, bazata pe notiunile de *principal* si *acces control list* (***ACL*** - în traducere lista de control a acceselor). În esenta, un *principal* se defineste a fi subiectul/entitatea ce actioneaza la un anumit moment asupra unei resurse din sistem, resursa aflata sub incidenta unei ***ACL***. ***ACL***-urile sunt concepute pentru a contine informatii despre permisiunile si restrictiile referitoare la *principal*-ii existenti în sistem.

Jini utilizeaza o schema de securitate practic identica cu cea din sistemul ***CORBA***. Se vede aici faptul ca Sun, proiectantul ***Jini***, a adus importante contributii si în proiectarea sistemului ***CORBA***.

KnittingFactory îsi bazeaza schema de securitate în mare parte pe facilitatea *sandbox* (în traducere cutie cu nisip, facilitate ce restrictioneaza posibilitatile în executia unui applet; un exemplu ar fi imposibilitatea conectarii acestuia oriunde în retea ci numai la calculatorul de unde a provenit – de unde a fost încarcat) oferita de Java pentru executia applet-urilor la distanta.

BOND ofera o schema de securitate bazata pe un server specializat, ce poate asigura autorizarea si autentificarea acceselor în sistem. În esenta însa, schema nu este activata implicit.

Infospheres implementeaza o schema de securitate la nivelul fiecarui inbox asociat unui obiect unde se gasesc doua liste de acces ce contin identificatori de obiecte ce au sau nu drepturi de acces asupra cozii de intrare a obiectului respectiv. Exista o lista ce contine obiectele ce vor fi rejectate în cazul unui acces si o a doua lista cu obiecte ce poseda permisiunea de acces asupra cozii.

Talking Objects nu implementeaza un mecanism de securitate în sine, ci utilizeaza serviciile de securitate oferite de sistemul Jet Virtual Medium. Acesta la rândul sau ofera mai multe scheme concurente de securitate. Una dintre ele se bazeaza pe existenta unui *plugin* de securitate (cod extern introdus în sistem), facând astfel posibila upgradarea rapida a sistemului în cazul aparitiei unor protocoale de securitate (criptare, compresie etc.) noi, mai performante.

Evenimente distribuite

Un concept de maxima importanta în orice sistem distribuit, îndeosebi în cadrul unor sisteme ce îsi propun caracteristici de scalabilitate extrema, îl constituie notiunea de ***eveniment distribuit***. Nu vom defini aici notiunea de *eveniment*, preferând adoptarea unei definiri intuitive, ce ar putea mai bine îngloba totalitatea semnificatiilor existente în diferitele implementari. Atributul *distribuit*, însa, defineste practic capacitatea structurii distribuite de a actiona ca un tot unitar. Vom remarca de asemenea ca, în majoritatea cazurilor, notiunea de eveniment va putea fi asociata notiunii de ***message-passing***, primirea unui mesaj fiind ea însasi un eveniment.

O alta problema ce apare în cazul evenimentelor distribuite, problema ce ar fi trebuit tratata separat, ce însa nu face obiectul prezentului studiu, este cea a mentinerii unei coerente temporale în orice sistem distribuit. În esenta, fara aceasta caracteristica *http://jvm.sunsite.pub.ro*, un sistem de prelucrare cu resurse de calcul distribuite nu poate converge, în activitatea sa, catre realizari coerente.

Putine dintre sistemele prezentate trateaza în mod explicit, notiunea de eveniment distribuit. Majoritatea însa permit implementarea unor scheme de "ascultare sau asteptare" (*listen-callback*) a aparitiei unor evenimente.

Exista doua abordari ce trateaza definirea legaturii dintre notiunea de eveniment distribuit si cea de mesaj. Un mesaj poate fi privit ca fiind el însusi un eveniment încapsulat. De asemenea, transmiterea si receptia mesajelor pot fi considerate declansatoare de evenimente.

Jini, prin utilizarea tehnologiei ***JavaSpaces*** [12] (deci si a evenimentelor distribuite definita de aceasta), a notiunii de *Listener*, implementeaza ambele scheme definitorii ale evenimentelor distribuite.

BOND specifica, prin obiectul *bondEvent* o interfata pentru definirea unor evenimente distribuite însa în esenta trateaza aceste obiecte ca simple mesaje.

Infrastructura de servicii.

O caracteristica importanta a sistemelor prezentate, este posibilitatea integrarii unor aplicatii heterogene precum si oferirea serviciilor acestora catre alte aplicatii în construirea (build-up) unor scheme (eventual ierarhice) de *servicing* distribuit.

Dupa ce s-a tratat cazul infrastructurii distribuite de comunicatie, fara de care nu ar fi posibila implementarea celei de servicii, utilizând facilitatile nivelului inferior (sistemul distribuit de comunicatie), se construieste o alta structura logistica distribuita, ce se ocupa de ceea ce reprezinta cu adevarat finalitatea sistemului în ansamblu, si anume structura de servicii si operatiile aferente.

Aceasta suprapunere de niveluri logice (*layering*) este esentiala si se regaseste în majoritatea arhitecturilor moderne. Spre exemplu însasi Internet-ul poate fi considerat un urias sistem distribuit, chiar si din perspectiva unui simplu apel al unei primitive de tip send(), prin implementarea unei transparente totale a procesului de transmisie efectiva (rutare) a datelor în retea. Un nivel superior de structura distribuita poate fi considerat World Wide Web, aceasta imensa intretesere de hipermedia, ce utilizeaza în esenta serviciile oferite de Internet.

Service

Service (serviciu oferit) este un termen extrem de utilizat în prezent, în domeniul sistemelor distribuite. Semnificatia sa însa, este extrem de variata, în functie de context si mai ales de implementare. Vom considera în definitia noastra o mare parte a elementelor comune, întâlnite în implementarile discutate.

Astfel, prin serviciu, vom întelege o notiune alternativa pentru cea de *procesare specifica, definita de un anumit algoritm*, urmând ca, în cazul în care discutam despre *oferirea* unui serviciu, sa ne referim la existenta unei facilitati de calcul ce executa algoritmul respectiv, iar în cazul în care discutam despre *cautarea* unui serviciu, sa ne referim la gasirea unei facilitati de calcul ca mai sus.

Exista de asemenea, în majoritatea sistemelor, un alt termen important, cel de *interfata serviciului* (service interface), interfata corespunzatoare unui anumit serviciu. Aceasta este o structura ce defineste în principal capabilitatile serviciului respectiv precum si elementele de interes pentru accesarea din exterior a acestora. De aceea în cazul particular al cautarii unui serviciu vom putea discuta de asemenea de cautarea unei interfete satisfacatoare pentru procesarea dorita.

Serviciul poate fi privit deci în esenta ca o transformare de tip functional (functie ce genereaza o iesire dintr-o anumite intrare), având asignate niste caracteristici aditionale ce definesc utilizarea acesteia.

În cadrul ***CORBA*** serviciul este definit ca fiind orice obiect înregistrat într-un anumit repository (pentru informatii despre repository vezi 2.2.3), obiect ale carui metode vor fi practic utilizate în accesarea capabilitatilor serviciului definit de acesta.

Jini defineste în mod explicit notiunea de serviciu ca fiind orice entitate activa în sistem, entitate ce ar putea eventual interactiona cu alte entitati. Serviciile sunt organizate în federatii, grupari logice ce pot defini eventual anumite grupuri de interes sau

calcule complexe. De remarcat ca prin deinitia data, ***Jini*** extinde definitia prezentata de noi mai sus.

În cadrul ***KnittingFactory*** serviciul poate fi regasit la nivelul fiecarui worker activ din sistem, worker ce îsi ofera practic puterea de calcul pentru a fi utilizata de catre managerul activ. Cazuri similare sunt ***Calypso*** si ***Charlotte***.

BOND ascunde notiunea de serviciu în cea de *agent*, prezentând caracteristici de sistem multi-agent.

Infospheres se prezinta cu o abordare inuitiva, asemanatoare cu cea din cadrul ***CORBA***. Orice aplicatie existenta la un moment dat în cadrul sistemului sub forma unui proces, aplicatie ce poate fi accesata de alte aplicatii este privita ca un potential participant la un *task force* (vezi 3.6 pentru definitia notiunii de *task force*).

Talking Objects utilizeaza la maxim notiunea de *service*. Astfel un *service* este definit ca fiind una dintre facilitatile oferite de catre un anumit *service provider* (ofertant de servicii) activ la un anumit moment în cadrul sistemului. *Service provider*-ul este o aplicatie ce contine mai multe module functionale (facilitati oferite), ce "asteapta" aparitia unui asa-numit *service request*. Un *service request* este în esenta o cerere de serviciu, formulata într-un limbaj specific (un subset al limbajului natural). Un alt element interesant în cadrul ***Talking Objects*** este posibilitatea integrarii unor aplicatii existente prin intermediul unui asa-numit *service activator*.

Bootstrap

Prin acest termen, în articolul de fata se întelege modalitatea prin care un sistem distribuit împreuna cu structura sa de servicing, sunt definite si "pornite" efectiv.

Aceasta modalitate poate avea un impact extrem de puternic asupra unor caracteristici importante ale sistemului, punând în discutie chiar transparenta acestuia. Pentru a fi mai expliciti, spre exemplu, functionarea nominala a unui sistem de servicing distribuit ce conserva transparenta devine total neinteresanta daca, pentru a putea accesa sistemul în ansamblu, va trebui sa cunoastem elemente netransparente, cum ar fi adresa IP a unuia dintre calculatoarele participante în sistem. În ciuda acestui fapt, spre surprinderea noastra, majoritatea sistemelor existente, incluzându-le chiar si pe cele mai sofisticate, nu trateaza aceasta problema în mod corespunzator.

Dintre toate sistemele prezentate, singurele ce ofera, conceptual, o transparenta totala chiar si la nivelul "pornirii" efective sunt ***Jini*** si ***Talking Objects***. Astfel Jini implementeaza notiunea de "ton de retea", ca fiind accesibila la orice socket ***Jini***-*compliant*, ramânând doar problema gasirii unui asemenea socket. ***Talking Objects*** pe de alta parte implementeaza un sistem de autodetectie atât a structurii de comunicatie existente cît si a serviciilor din sistem.

Lookup, Implementation Repositories, Registry

Lookup-ul este cea mai importanta faciltate (sau posibilitate, în anumite sisteme) ce trebuie sa existe în cadrul unui sistem de servicing distribuit. *Lookup*-ul (uneori numit si *discovery*) ofera practic posibilitatea "gasirii" unui anumit serviciu activ, existent la un anumit moment în cadrul sistemului, serviciu ce eventual se va dori accesat ulterior.

Aceasta "gasire" este un proces implementat în cele mai variate feluri în diferitele sisteme. Una din principalele probleme ale acestuia este pastrarea caracteristicii de sistem distribuit, si în special a transparentei aferente, (si nu multe sisteme reusesc acest lucru) precum si specificarea caracteristicilor serviciului cautat, specificare ce poate impune, într-o implementare slaba, limite extrem de neconvenabile în proiectarea unei aplicatii distribuite. Astfel cautarea unui serviciu se face în majoritatea cazurilor, prin specificarea unor caracteristici dorite ale acestuia însa însasi aceasta specificare este obiectul unor implementari din cele mai diverse. Unele din-

tre acestea permit specificarea interfetei serviciului dorit, întorcând referinte la unul sau mai multe servicii ce satisfac interfata respectiva, altele permit exprimarea functionalitatii dorite, în termenii unui limbaj special de definire a serviciului, iar alte implementari ofera chiar posibilitatea descrierii într-o exprimare pseudo-naturala (***Talking Objects*** [5], [6]).

Un alt element important, legat de lookup îl constituie stocarea fizica a serviciilor în sine, stocare definita de termenul *implementation repositories* (în traducere depozit de implementari). Anumite sisteme [2] implementeaza efectiv anumite structuri de stocare ce vor include în principal, implementarile serviciilor existente la un moment dat în cadrul sistemului. Desigur ca aceasta abordare prezinta multiple dezavantaje si de aceea este considerata învechita în multe din implementarile cu un design moderne. (Unele dintre aceste dezavantaje sunt generate de existenta la baza sistemului distribuit, a unei structuri mai mult sau mai putin centralizate.) În continuare sunt analizate anumite implementari si solutii gasite pentru problema stocarii efective a codului definitoriu pentru serviciile active.

În ***CORBA***, existenta unor Interface *Repositories* precum si a unor servicii de *Registry* (înregistrare a serviciilor la intrarea în sistem), *Naming*, *Trading* etc. ofera multiple facilitati de *lookup*. Astfel se pot cauta servicii pe baza specificarii unor proprietati ale acestora, pe baza specificarii interfetelor obiectelor si/sau metodelor aferente sau chiar pe specificarea unor caracteristici de baza ale interfetelor. Exista de asemenea un limbaj de definire a interfetelor, limbaj ce actioneaza ca un standard atât în interiorul sistemului, cât si în interfatarea acestuia cu alte sisteme similare externe.

Jini rezolva problema *lookup*-ului într-un mod similar prin implementarea unei facilitati de *registry* combinata cu un mecanism de *lookup*. Un mecanism interesant este cel definit prin notiunea de *leasing* (pentru detalii a se vedea 2.2.4).

KnittingFactory nu face exceptie de la regula. Din nou un *registry* si un mecanism de *lookup*. Partea interesanta este implementarea efectiva a acestuia, realizata cu ajutorul unei tehnologii extrem de simple (însa cu grave probleme de scalabilitate), folosind HTTP, HTML si Javascript.

În cadrul ***Calypso***, notiunea de lookup este practic înlocuita prin vointa fiecarui *worker* (participant activ în sistem) de a-si manifesta dorinta de a participa într-o anumita structura de calcul. Acest lucru se realizeaza prin efectuarea unui broadcast/multicast, ce va fi preluat de catre un asa-numit manager de calcul, manager efectuand calculul respectiv.

BOND, ca si ***Jini*** sau ***CORBA***, ofera o facilitate de directory prin intermediul unui asa numit *directory server*, desemnând o autoritate centrala.

Talking Objects abordeaza problema dintr-o perspectiva complet diferita. Gasirea precum si compunerea serviciilor în cadrul sistemului se bazeaza în mare masura pe gruparea anumitor *service provider*-i pe canale specifice, canale definite în principal de elementele importante din descrierea în limbaj natural a capabilitatilor oferite. Astfel gasirea unui serviciu se reduce în esenta în efectuarea unui broadcast optimizat (utilizând serviciul multicast din ***Jet Virtual Medium*** [6]) în interiorul grupului de interes (multicast). De remarcat ca un anumit *service* (si chiar un anumit *service provider*) pot fi simultan active în mai multe grupuri distincte.

Integrarea, compunerea serviciilor. Accesarea unui serviciu.

Dupa obtinerea unei referinte distribuite la un serviciu activ existent, stocat într-un *repository* îndepartat (necunoscut), se pune problema utilizarii efective a acestuia. În acest paragraf va fi analizata aceasta problema.

De asemenea, ar fi de preferat obtinerea unui maxim de eficienta din utilizarea sis-

temului, prin integrarea (eventual în *lookup*) a unei modalitati de compunere eficienta a mai multor servicii existente, pentru oferirea unui serviciu complex, capabil sa rezolve cât mai mult din algoritmul beneficiar (calculul în beneficiul caruia se acceseaza serviciile respective). Aceasta eficientizare ar putea fi comparata cu optimizarea unor cereri succesive în cadrul unui sistem de baze de date de tip *data warehouse*.

CORBA ofera multiple posibilitati de acces la obiectele existente. Majoritatea acestora sunt însa bazate pe solutia *stub-skeleton*, în care un client va transfera, prin intermediul *ORB*-ului, o parte a codului necesara în comunicarea cu obiectul original, aflat pe server. Aceasta schema se regaseste de asemenea si în cadrul ***BOND***, unde este utilizata notiunea de *shadow* (umbra), un obiect accesibil la nivelul clientului, obiect ce poate fi utilizat pentru accesarea unui alt obiect (obiectul original), aflat la distanta. Un alt element interesant gasit în cadrul sistemului ***BOND*** îl constituie utilizarea unui limbaj descriptiv de comunicatie interagenti [13], limbaj numit ***KQML***.

Jini [11] extinde modelul ***CORBA/BOND/RMI*** prin virtualizarea accesului si introducerea unor notiuni noi cum ar fi *leasing* sau *service-proxy*. Prin *leasing*, un anumit serviciu îsi poate asigura accesul asupra altuia pentru o anumita perioada de timp. *Service-proxy*-ul reprezinta o idee similara cu cea de *stub-skeleton* sau *shadow* însa la un nivel logic superior.

În ***KnittingFactory***, accesarea serviciilor înregistrate se poate face utilizând în principal tehnologia ***JavaRMI*** [14].

Calypso si ***Charlotte*** ofera accesul la resursele sistemului într-un mod transparent, la nivelul limbajului de programare. La un moment dat, pornirea unui nou calcul paralel distribuit implica executarea unei aplicatii principale (managerul) scrisa într-un limbaj special extins (extensie de C sau Java), precum si existenta unor procese worker dispuse sa participe la calculul distribuit. Acestia se anunta periodic, cooperând cu managerul în distribuirea calculului respectiv.

Talking Objects [5] implementeaza un mecanism ce asigura un maxim de transparenta, prin utilizarea expresiilor în limbaj natural. Astfel, o entitate va putea accesa orice serviciu, în urma formularii în limbaj pseudo-natural, a capabilitatilor necesare la un anumit pas (si efectuarii ulterioare a unui multicast în grupul de interes corespunzator, a se vedea si 2.2.3), urmata de selectia unui service-provider dintre cei ce au raspuns cerintelor. Un anumit nivel calitativ asociat fiecarui serviciu va determina selectia *service provider*-ului ce va efectua calculul/serviciul respectiv, urmând alegerea celui mai apropiat sau calitativ superior.

Implementari – particularitati

În continuare vom încerca detalierea unor caracteristici originale, definitorii pentru fiecare sistem în parte.

CORBA

CORBA este un efort unificat al mai multor companii, menit sa contracareze în principal evolutia Microsoft prin sistemul Windows95/98 ce îsi propunea din ce în ce mai multe teluri ce ar duce în final la construirea unui urias sistem distribuit, bazat pe o solutie Windows în conjunctie cu tehnologia Internet. Principalii contributori la CORBA sunt: BNR Europe Ltd, Expersoft Corporation, IBM Corporation, ICL plc, IONA Technologies Ltd, Digital Equipment Corporation, Hewlett-Packard Company, HyperDesk Corporation, NCR Corporation, Novell USG, Object Design, Inc, Sun Microsystems Inc, SunSoft, Inc etc. O referinta online la CORBA este ***http://www. omg.com***.

În prezent unul din actorii principali pe scena ***CORBA***, Sun Microsystems s-a distantat considerabil prin introducerea noului sau prototip, ***Jini***, bazat pe tehnologia Java.

Jini

Jini este (în prezent) un prototip dezvoltat de cei de la Sun Microsystems, prin departamentul JavaSoft. O referinta online la ***Jini*** este ***http://www.javasoft.com/products/jini***. ***Jini*** prezinta (cel putin în versiunea initiala multiple probleme nerezolvate, una dintre cele mai importante fiind scalabilitatea sistemului în ansamblu, caracteristica ce ar trebui implementata extrem de eficient pentru a atinge dimensiunile pretinse ale sistemului.

De asemenea ***Jini*** este bazat în totalitate pe solutii si tehnologii Java, incluzând ***JavaSpaces***, ***JavaRMI*** etc., tehnologii ce sunt înca detinute de Sun Microsystems, cu toate concesiile facute în acest sens.

Jini îsi propune preluarea sarcinii integrarii unei multitudini de servicii (hardware inteligent, aplicatii software, utilizatori umani) sub forma unor entitati virtuale într-un urias mediu distribuit, construit utilizând tehnologie Java, medii de comunicatie din cele mai variate, bazându-se mai ales pe suportul a sute de producatori de echipamente cu utilizare de masa ce vor fi adaptate în viitor pentru a deveni ***Jini***-*compliant*. Succesul acestui început temerar sta desigur în puterea conceptiei initiale precum si a convergentei acesteia în cazul scalarii prototipului sistemului la un sistem real compus din milioane de entitati. Ceea ce poate functiona pentru un prototip compus din 10-20 de calculatoare si dispozitive, poate sa nu mai fie functional în cazul a sute de mii sau chiar milioane de entitati active, cum îsi propune sistemul.

Aceasta viziune va trebui sustinuta si de caracteristicile mediului global constituit din masini interconectate ce vor rula codul distribuit. Desi se presupune existenta unor medii de comunicatie de viteze diferite cu protocoale diferite, a crede ca rularea unui sistem Jini în retele de tip TCP/IP bazate pe linii de comunicatie seriala sub 128-256Kbps este desigur o utopie. Pornind de la început cu stacheta suficient de sus se pot evita complicatii în proiectarea sistemului. Se presupune deci existenta unei retele bazate pe medii de viteze de peste 10Mbps precum si a unei latente mici, de ordinul secundelor cel mult. Astfel iese din discutie viabilitatea unei implementari ***Jini*** bazate pe Internet în ansamblul sau, cel putin în viitorul apropiat. Mediile cele mai adecvate acestui scop sunt desigur cele de mare viteza de tipul infrastructurii digital-tv sau ATM precum si a echipamentelor de tip bancomat, tv-on-demand etc.

De asemenea exista prezumtia implicita ca orice hardware ***Jini*** conectat în sistem poseda suficienta memorie precum si o anumita putere de procesare elementara însa suficient de complexa pentru a suporta macar compatibilitate cu un sistem înconjurator în mare parte Java. Chiar si perifericele simple (de exemplu cele de stocare de date) vor trebui sa fie controlate de catre un cod activ ce va implementa interfata acestora cu reteaua în ansamblu. Acest lucru presupune desigur reproiectarea multora dintre produsele existente în prezent pentru a crea niste versiuni *Jini-compliant*.

Din punct de vedere arhitectural, ***Jini*** devine mai tot complex pe zi ce trece, facilitatile de migrare si mesagerie distribuita a obiectelor (*Object Serialization, Remote Method Invocation*), modelul evenimentelor si tranzactiilor asincrone (*Distributed Event Specification, Transaction Specification*) fiind completate de tehnologie de integrare si identificare distribuita a obiectelor si codului activ (***JavaSpaces***) precum si de optimizarile de cod necesare pentru a putea rula cod eficient în medii sarace în resurse (memorie putina, procesor lent, putere mica) cum ar fi masina de spalat sau frigiderul de exemplu.

Afirmatia ca ***Jini*** reprezinta deci un pas înainte, depasind toate dificultatile, constituind un urias sistem distribuit, ce înglobeaza practic o infinitate de entitati, ne uimeste si ne incita în acelasi timp. Specificatiile **Jini**, desi deocamdata extrem de sumare, par sa sustina aceasta afirmatie chiar

si în acest stagiu incipient. Sa fie oare Jini urmatoarea mare bomba ce ne apropie din ce în ce mai mult de integrarea informatica universala? Oricum ar fi, este un pas important înainte, oferind imense posibilitati în domeniul design-ului sistemelor distribuite ce vor putea astfel integra milioane de entitati software, hardware sau logistice.

KnittingFactory

KnittingFactory (în traducere *fabrica de tricotaje*) este o realizare a unei echipe de la New York University, precursoare a sistemului Jini. Telurile propuse sunt mai realiste decât ale altor sisteme (vezi ***Jini***) gasindu-si în final si o implementare functionala.

Elemente originale ale ***KnittingFactory*** ar fi cele legate în principal de realizarea prelucrarilor efective în interiorul unor appleti Java, localizati într-un browser-client distant.

Calypso & Charlotte

Calypso împreuna cu implementarea sa Java - ***Charlotte*** (desi nu este specificat explicit acest lucru, modelul arhitectural al celor doua sisteme este practic identic), sunt sisteme dezvoltate de echipa ce a dezvoltat în mare parte si sistemul ***Knitting Factory***.

O mare diferenta a acestor doua sisteme fata de celelalte este faptul ca nu îsi propun relizarea unei largi structuri distribuite, ci mai degraba a unei structuri de prelucrare paralela, în cadrul unui sistem intranet cu resurse de comunicatie (rata de transfer etc.) mult mai mari decat în cazul unui mediu de comunicatie bazat pe Internet.

BOND

BOND [9], un sistem dezvoltat la Purdue University, se încadreaza în generatia noua de sisteme bazate pe tehnologie portabila (de exemplu Java) si pe un mediu de comunicatie Internet, cum ar fi ***Talking Objects***, ***InfoSpheres*** sau ***Jini***. Referinte online la ***BOND*** se pot gasi la ***http://bond.cs. purdue.edu***.

Un element extrem de interesant ce se gaseste si în ***BOND*** se afla în implementarea mecanismului de distribuire efectiva a entitatilor si obiectelor. Mecanismul numit *shadowing* (în traducere *umbrire*) este în mare parte similar cu *proxy*-ul din ***Jini***. În versiunea prezenta, ***BOND*** utilizeaza primitivele de comunicatie existente în sistemul ***InfoSpheres***.

InfoSpheres

InfoSpheres [10], avându-si originile în chiar zilele de început ale tehnologiei Java, la California Institute of Technology, porneste de la cu totul alte premise decât cele uzual întâlnite în cadrul celorlalte sisteme similare existente. Fiind finantat de US Air Force, proiectul atinge domenii legate în principal de logistica si automatizarea procesului de constituire a grupurilor de solutionare a task-urilor (*task force*).

Astfel un *task force* se defineste a fi o structura de procesare heterogena, cu un timp de existenta variind de la cîteva secunde la cîteva ore. În esenta se încearca modelarea notiunii de task force dintr-un mediu real prin structuri logice, implementate în contextul unui sistem distribuit.

Ideea si abordarea îsi propun în esenta satisfacerea sponsorului principal, de natura militara, prin elaborarea unor mecanisme ce automatizeaza crearea rapida a unui task force la aparitia unui eveniment ce urmeaza a fi tratat. Este vorba deci, despre compunerea unor servicii distribuite într-o structura convergenta, capabila de rezolvarea problemei propuse.

Modalitatile de realizare si implementare a proiectului include elemente similare cu ale multor alte sisteme de acest tip. Principala implementare a sistemului ofera de asemenea o eficienta infrastructura distribuita de comunicatie, ale carei servicii sunt utilizate în multe alte proiecte de calcul distribuit. Unul dintre acestea ar fi ***BOND***.

Informatii online despre ***InfoSpheres*** cât si despre aplicatii utilizând sistemul în prezent, sunt accesibile la ***http://www.infospheres. caltech.edu***.

Talking Objects

Talking Objects [5], un sistem dezvoltat de noi, precedând ***Jini*** cu aproape 2 ani, îsi propune oferirea unor facilitati apropiate de cele prezentate de Jini însa într-o modalitate noua, încercând o rezolvare eficienta si realista a problemei scalabilitatii cu care ***Jini*** se confrunta înca din faza de proiectare. ***Talking Objects*** precum si sistemul de comunicatie utilizat, ***Jet Virtual Medium*** [6], sunt accesibile online la ***http://talking.sunsite.pub.ro*** respectiv ***http ://jvm.sunsite.pub.ro***.

Una dintre ideile originale promovate de ***Talking Objects*** este utilizarea unui limbaj pseudo-natural în comunicatia aferenta facilitatilor de servicing. Mai exact, proiectarea unui subset al limbajului natural, subset ce va fi ulterior utilizat în exprimarea primitivelor de servicing, atât a requesturilor cât si a celor aflate în interiorul oricarui *service-provider* activ. Pe scurt, combinarea unui subset de limbaj natural, cunoscut tuturor entitatilor active în sistem, cu un algoritm distribuit de extindere ulterioara a acestui limbaj, prin constructii semantice, ofera o distributie maxima, în contextul unei structuri Internet, având potentialul includerii unui numar extrem de mare de entitati active.

Sistemul îsi propune de asemenea integrarea aplicatiilor existente prin intermediul unor agenti numiti *service-activator*, agenti ce traduc facilitatile intrinseci ale aplicatiei într-o descriere pseudo-naturala iar apoi activeaza serviciul respectiv, în cadrul sistemului.

Sistemul introduce multe alte noutati, una dintre ele fiind o extensie a notiunii de Resource Locator (URN, URL) pentru a include si alte clase de obiecte distribuite accesibile în sistem.

Concluzii. Intrebari deschise.

Sistemele discutate în aceasta lucrare prezinta unele caracteristici ce suscita înca multe discutii, precum scalabilitatea sau portabilitatea.

Scalabilitatea este, si va fi pentru înca o perioada lunga de timp, o caracteristica pe cât de greu de atins pe atât de controversata. Discutia teoretica despre sisteme formate din mii sau chiar milioane de entitati, precum si gasirea unor solutii aparent corecte se confrunta cu realitatea implementarilor practice. Nici unul dintre sistemele descrise, în versiunile existente în prezent, nu rezolva problema scalabilitatii. Ba mai mult, unele dintre cele mai importante, trateaza aceasta caracteristica total inadecvat, punând sub semnul întrebarii însasi coerenta proiectarii sistemului în totalitatea sa.

O alta chestiune interesanta este derivata din dorinta obtinerii unei distributii maxime pentru a anumita arhitectura. ***Portabilitatea***, privita ca fiind o extensie a compatibilitatii software între platformele ce formeaza structura logistica a sistemului distribuit, asigura într-o mare masura posibilitatea unui înalt grad de distributie. Pe de alta parte, pretul platit trebuie cautat în viteza de procesare scazuta ca si în problemele de implementare greu de rezolvat.

O alta solutie ar fi combinatia dintre standardizarea/compatibilizarea interfetelor de comunicatie si acces la servicii, combinata cu translatoare rapide de cod pentru implementarea cazului mai rar, al migrarii totale inter-platforma. În acest ultim caz, entitatile ar trebui convertite între cele doua implementari, macar din punctul de vedere al codului executabil, daca nu si ca logistica.

Sistemele prezentate în lucrare au un impact mai mic sau mai mare în domeniul solutionarii unor probleme elementare înca. Finalitatea ce va fi perceputa peste multi ani poate, este însa cea care constituie motorul tuturor visurilor ale software designer-ului modern.

Autorul doreste sa multumeasca domnului Prof. dr. ing. Marian Dobre pentru sprijinul acordat în redactarea acestui material.

Bibliografie

[1] Baratloo A. , M. Karaul, H. Karl and Z.M. Kedem, KnitingFactory: An Infrastructure for Distributed Web Applications TR 1997-748, New York University, November 1997

[2] The Object Management Group & others, The Common Object Request Broker: Architecture and Specification, Revision 2.0 July 1995

[3] Baratloo A., P. Dasgupta, Z. M. Kedem, Calypso: A Novel Software System for Fault-Tolerant Parallel Processing on Distributed Platforms, 4th IEEE Symp. On HPDC, 1995

[4] Baratloo A., M. Karaul, Z. Kedem, P. Wyckoff, Charlotte: Metacomputing on the Web, 9th Intl. Conference on Parallel Distributed Computing Systems, 1996

[5] Radu Sion, Talking Objects: Distributed Servicing on the Internet, Politehnica University of Bucharest, Computer Science Presentation Session, 1999, http://talking.sunsite.pub.ro

[6] Radu Sion, Jet Virtual Medium: Distributed Communication on the Internet, http://jvm.sunsite.pub.ro

[7] Pradeep K. Sinha, Distributed Operating Systems – Concepts and Design, IEEE Computer Society Press, 1997

[8] A. S. Tanenbaum, Distributed Operating Systems, Prentice Hall, 1995

[9] D. Marinescu and others, BOND, Purdue University, http://bond.cs.purdue.edu

[10] K. Mani Chandy, InfoSpheres: Information Infrastructure for Task Forces, California Institute of Technology, http://www.infospheres.caltech.edu, November 1996

[11] Jim Waldo, Jini Architecture Overview, © 1998 Sun Microsystems

[12] JavaSpaces Specification, © 1998 Sun Microsystems, http://java.sun.com

[13] Tim Finin, Richards Fritzon, Don McKay, Robin McEntire, KQML as an Agent Communication Language, Proceedings of The Third International Conference of Information and Knowledge Management, ACM Press, November 1994
Other KQML related docs.

[14] The Java Remote Method Invocation, Java Development Kit, © Sun Microsystems, http://java.sun.com