



RAPORT INTERMEDIAR DE ACTIVITATE (RIA)

Denumirea Proiectului: “GEEA – Centru de resurse GRID multi-corE de înaltă pErformAnță pentru suportul cercetării, dezvoltării tehnologice și inovării științifice pe plan European”

Contract nr: 51/11.05.2009

Sursa de finanțare: POS CCE O2.2.3

Data finalizării: 10.05.2011

Cuprins

Activitatea 3.3. Stabilirea priorităților în raport cu nevoile specialiștilor și utilizatorilor sistemului GRID național	2
Parteneri și utilizatori ai sistemelor Grid	2
Direcții de cercetare identificate.....	5
Activitatea 3.1. Proiectarea aplicațiilor ce vor fi dezvoltate prin explorarea proiectelor similare la nivel european și internațional, pentru a asigura interoperabilitatea	8
Monitorizarea în sistemele distribuite.....	8
Soluții existente de monitorizare pentru sisteme distribuite.....	8
Descrierea soluției de monitorizare propuse	9
Simularea sistemelor distribuite.....	13
Planificarea în sisteme distribuite	16
Servicii Grid (implementarea SOA pentru Grid).....	20
Bibliografie.....	24



GEEA – Centru de resurse GRID multi-corE de înaltă pErformAnță pentru suportul cercetării, dezvoltării tehnologice și inovării științifice pe plan European

Activitatea 3.3. Stabilirea priorităților în raport cu nevoile specialiștilor și utilizatorilor sistemului GRID național

Activitatea 3 a proiectului GEEA este realizarea de aplicații informatice necesare pentru administrarea și funcționarea centrului de resurse Grid al Universității Politehnica din București (UPB) și pentru optimizarea funcționării Grid-ului la nivel național. În cadrul acestei activități, în prima fază au fost proiectate aplicațiile ce vor fi dezvoltate prin explorarea proiectelor similare la nivel european și internațional pentru a asigura interoperabilitatea cu acestea. Utilizarea capacităților locale de a dezvolta aplicații este importantă pentru stabilirea priorităților în raport cu nevoile specialiștilor și utilizatorilor sistemului GRID național. Activitatea va consta în fazele următoare în dezvoltarea aplicațiilor identificate în raport cu prioritățile identificate și capacitățile stabilite precum și testarea aplicațiilor și realizarea documentațiilor de utilizare. De asemenea stabilirea continuă a capacităților de a dezvolta aplicații va conduce la posibilitatea de creare de colaborări pe componenta de dezvoltare de aplicații. Prin stabilirea capacității de a dezvolta aplicații se asigură și perfecționarea echipei existente.

În vederea stabilirii priorităților de cercetare pentru proiectul GEEA, au fost analizate împreună cu principalii parteneri și beneficiari ai proiectului direcțiile principale evoluției a sistemelor de tip Grid și a aplicațiilor dedicate. Principalii beneficiari ai proiectului sunt: Consorțiul RoGrid, Facultatea de Automatică și Calculatoare din cadrul Universității „Politehnica” din București, Oficiul pentru Administrare și Operare al Infrastructurii de Comunicații de Date RoEduNet.

Obiectivele principale ale proiectului GEEA sunt corelate cu necesitățile la nivel național cu privire la sistemele Grid. UPB are numeroase parteneriate în proiectele bazate pe sistemul GRID, cu instituții care au participat inițial la constituirea consorțiului RoGrid.

Stabilirea priorităților în raport cu nevoile specialiștilor și utilizatorilor sistemului GRID național au fost realizate în colaborare cu principalii parteneri utilizatori de Grid din România.

Parteneri și utilizatori ai sistemelor Grid

Institutul Național de Cercetare – Dezvoltare în Informatică (ICI), este coordonatorul a numeroase proiecte de cercetare, printre care din domeniul Grid-ului amintim: Virtual organization based on GRID technology for high performance modelling, simulation and optimization. ICI are o tradiție și o prezență activă de peste 35 de ani în informatica românească, reprezintă cel mai important institut de cercetare-dezvoltare și inovare în domeniul Tehnologiilor Informației și Comunicațiilor din România. Prin asumarea priorităților științifice și tehnologice ale domeniului, misiunea ICI o constituie consolidarea poziției

câștigat și dezvoltarea suportului științific și tehnologic în domeniul tehnologiilor informației și comunicațiilor necesar realizării structurilor și serviciilor specifice societății informaționale bazate pe cunoaștere. Arii de competență ale ICI sunt orientate în domenii precum: rețele de comunicație și tehnologii avansate pentru dezvoltarea de aplicații în medii distribuite; tehnologiile informației și comunicațiilor în domenii de interes public; conținut digital, creativitate și dezvoltare personală; tehnologiile informației și comunicațiilor, suport pentru dezvoltarea afacerilor și a industriei; noi modele și tehnologii în ingineria sistemelor și a produselor software; sisteme bazate pe cunoștințe, învățare și sisteme cognitive; sisteme avansate de calcul și control. Portofoliul de produse create în Institut sau în colaborare cu parteneri interni sau externi constă din: aplicații de bioinformatică, modele avansate de decizie, modele de business, e-business, e-commerce, modele și sisteme de diagnoză și evaluarea riscului, modele, algoritmi și tehnologii GRID, platformă GRID, platforme pentru promovarea afacerilor electronice, produse software pentru educație asistată de calculator.

Pagina web: <http://www.ici.ro/>

Institutul Național de Fizica a Pământului (INFP). Principalele activități ale INFP sunt legate de monitorizarea seismică. Rețeaua Seismica Națională are în componență 73 de stații seismice digitale în timp real (499 de canale la 100 eşantioane / canal) instalate între 1995 până în prezent. Stațiile seismice sunt echipate cu digitizoare Q330 și K2, senzori de accelerație (EpiSensor, 2g) și senzori de viteză (de bandă largă și scurtă perioadă). Cele mai multe dintre stații sunt amplasate în estul și sudul Carpaților fiind destinate în primul rând monitorizării seismice a zonei Vrancea. Transmiterea de date dintre stațiile seismice digitale și Centrul Național de date de la București se face prin diferite sisteme de comunicație (GPRS, VPN, satelit, radio, linii închiriate și Internet), astfel încât back-up în cadrul rețelei s-a realizat prin distribuirea sistemelor de comunicație. Datele digitale de la stațiile seismice sunt analizate în timp real de un sistem automat de achiziție și prelucrare automată. Rețeaua Seismica Națională distribuie rezultatele sale (localizări și fazele cutremurelor), prin Internet, către mai multe servicii seismologice din Europa.

O altă direcție importantă se referă la sistemele complexe de monitorizare a Pământului. Sistemele complexe de monitorizare și prelucrare în timp real și cvasi-real sunt destinate urmării factorilor precursori ai evenimentelor seismice, fapt care conduce la alcătuirea unei bănci de date ce poate fi analizată în paralel cu activitatea seismică în vederea elaborării unei metode eficiente de prognoza pe termen scurt a activității regiunii Vrancea. Factorii precursori avuți în vedere sunt: activitatea seismică în domeniul micro-cutremurelor, variația câmpului magnetoteluric corelată cu activitatea solară, variația concentrației gazelor alfa radioactive în lungul faliilor, ionizarea atmosferică în zona centrală, măsurători de emisie acustică în foraje, variația curenților telurici, variația nivelului pânzei freatice precum și alte fenomene și observații considerate a avea legătura cu activitatea seismică. Pentru fiecare dintre factorii enumerați mai sus INFP a realizat un sistem avansat și inteligent de detecție, de procesare și de analiză a datelor experimentale utilizându-se o bază comună de timp și echipamente compatibile cu cele existente în Rețeaua Națională de supraveghere seismică.

Proiectele de cercetare în care institutul este implicat sunt: Cercetări fundamentale de seismologie istorică și paleo-seismologie necesare pentru evaluarea seismicității pe termen lung și a hazardului seismic, Macro-zonarea seismică a teritoriului României, fundamentată pe intensități macro-seismice reevaluate coroborate cu date geologice și geofizice complexe, Reducerea riscului seismic prin evitarea rezonanței teren-structură și prin izolarea bazei structurii. Aplicabilitate în zona metropolitană București, Modelarea efectelor seismice locale induse de cutremurele crustale produse în zonele Tulcea, Galați - Tecuci, Câmpulung, Banat și evaluarea riscului seismic al acestor zone.

Pagina web: <http://www.infp.ro/>

Institutul de Fizica Chimie "Ilie Murgulescu" (ICF). Domeniile de activitate ale ICF sunt Cercetare, dezvoltare, proiecte, asistență, consultanță, experize privind metalurgia extractivă a metalelor neferoase și rare, aliajelor, pulberilor, materialelor ceramice și compozite, stocarea și prelucrarea subproduselor și deșeurilor solide, valorificarea metalelor neferoase și rare. Proiectele principale ale institutului sunt: Monte Carlo calculations regarding the evaluation of the magnetic properties of the 'core-shell' type nanocomposite materials, An EPR study of the synthesis, products and post-synthesis treatments using spin-labeled components, Contributions concerning the alloying element effects on the stability of a new industrial alloy in very aggressive environments, Hybrid catalysts with oxo(peroxo) metal species obtained by functionalization of mesostructured silica for selective oxidation of olefins, The influence of microstructure and composition on the thermodynamic behaviour of some micro and nanostructure multicomponent oxide materials with special properties, Fractal characterization of Pd deposition on carbon nanotubes and Au(hkl).

Pagina web: <http://www.icf.ro/>.

Institutul National de Cercetări Aeronautice "Elie Carafoli" (INCAS). INCAS este singura unitate din România care desfășoară activitate de cercetare pe întregul ciclu, pornind de la cercetarea fundamentală de bază și orientată, continuând cu cercetarea aplicativă și terminând cu dezvoltarea tehnologică și implementarea rezultatelor obținute în producție. Cercetarea fundamentală pe care o desfășoară INCAS urmărește creșterea nivelului de cunoștințe în domeniul științelor aeronautice și aerospațiale, referindu-se la: aerodinamica generală, dinamica zborului și sistemelor, structuri aeronautice, aeroelasticitate, rezistența materialelor utilizabile în aeronautică, sisteme de propulsie aeronautice. Cercetarea aplicativă - dezvoltarea tehnologică, care intră în profilul institutului, se referă la elaborarea de: tehnologii și materiale aeronautice; echipamente electronice, mecano-hidraulice și pneumatice; realizarea de modele experimentale din domeniul aeronautic și aerospațial; standuri și instalații de testare; platforme și stații pilot; aparatura de laborator, scule și dispozitive utilizabile în industria aeronautică. Proiectele acestui institut sunt: Dezvoltarea Mock-UP virtual și realizare specimen proba statică pentru aeronava de transport aerian regional, Sistem sinergic pentru lansarea/recuperarea dinamică a obiectelor satelitare - SILROS, Structuri Morphing pentru aplicații în construcții Aeronautice - SMART, Materiale avansate pentru industria aeronautică: Compozite C-C cu matrice mezofazică aditivă cu nano-tuburi de carbon și compozite laminate fibre - metal COMPAS.

Pagina web: <http://www.incas.ro/>

Administrația Națională de Meteorologie (ANM). Obiectul principal de activitate al Administrației Naționale de Meteorologie îl constituie activitatea de meteorologie și climatologie necesara dezvoltării social-economice a României și integrarea acestei activități în sistemul de convenții și relații internaționale, în care scop: efectuează observații și măsurători asupra stării și evoluției vremii; asigura și controlează circulația internă de date și informații meteorologice; elaborează metodologii de măsurare și prelucrare a datelor și realizează produse meteorologice; elaborează diagnoze, prognoze și avertizări meteorologice; realizează studii de cercetări în domeniile meteorologiei; expertizează, validează și stochează date și informații specifice Fondului național de date meteorologice, în vederea tinerii la zi și constituirii băncilor de date tematice la nivel național; elaborează studii și scenarii de evoluție a climei; asigura schimbul de date și de informații meteorologice în flux rapid; realizează schimbului internațional de date în domeniul meteorologiei; elaborează componentele de climatologie din studiile de impact și de bilanț de mediu; organizează și coordonează sistemul național de formare și pregătire profesională în domeniul meteorologiei, climatologiei și fizicii atmosferei. O componenta importanta a activității de meteorologie o constituie prelevarea și transmiterea datelor de observații din rețeaua de stații de suprafața, radar și aerologice. Prin natura sa, meteorologia impune o strânsă colaborare internațională, Romania fiind membru fondator al Organizației Meteorologice Mondiale (OMM). Pe lângă schimbul extern de date, Administrația Națională de Meteorologie participa la acțiuni internaționale din cadrul programelor OMM, fie legate de proiecte științifice finanțate de Comunitatea Europeană (CE) sau colaborări bilaterale și multilaterale. Printre proiectele în care este implicată administrația națională de meteorologie amintim: cercetării climatologice, Fizicii Atmosferei și Poluării Aerului, Meteorologie Dinamica, Climatologie și Agrometeorologie, Metodică meteo, Modelare numerică, Prognoză de lungă durată, Teledetectie.

Pagina web: <http://www.meteoromania.ro/>

Direcții de cercetare identificate

Obiectivul 2 al proiectului GEEA constă în suportul aplicațiilor inovative ale instituțiilor de cercetare naționale pentru creșterea performanței și accesibilității sistemului național GRID și a centrului de resurse GRID local. Pe de o parte un rol important îl au aplicațiile pentru suportul asigurării interoperabilității, accesibilității, pentru monitorizarea și controlul sistemelor GRID, pentru utilizarea și partajarea optimă a resurselor. Aceste aplicații asigură suportul pentru un sistem distribuit de mari dimensiuni de încredere care oferă suportul aplicațiilor complexe și susținerea calculului de înaltă performanță. Pe de altă parte, suportul pentru aplicațiile dezvoltate de centrele de cercetare locale descrise în secțiunea anterioară (Institutul de Fizica Chimie "Ilie Murgulescu", Institutul Național de Cercetări Aeronautice "Elie Carafoli", Administrația Națională de Meteorologie etc.) oferă posibilitatea dezvoltării

acestora pe Grid, folosind un set de resurse de înaltă performanță. Sunt aduse astfel contribuții la obiectivul competiției de dezvoltare a sistemului Grid național prin soluțiile oferite pentru sporirea performanțelor.

Activitatea 3 susține realizarea obiectivului 2 al proiectului prin dezvoltarea de aplicații relevante pentru optimizarea funcționării sistemului Grid național prin aplicații de monitorizare a sistemului Grid creat, aplicații de simulare a sistemelor Grid în vederea optimizării aplicațiilor, aplicații de planificare a aplicațiilor în vederea optimizării timpului de acces la resurse și a echilibrării încărcării resurselor, aplicații pentru controlul și toleranța la defecte a sistemului Grid, publicarea pe site-ul proiectului a unui set de ghiduri de utilizarea a aplicațiilor create, ghiduri utile dezvoltatorilor de aplicații de bază. S-au identificat astfel următoarele direcții de cercetare și dezvoltare (direcții ce vor fi descrise în secțiunea următoare):

Utilizarea unui sistem de monitorizare pentru a îmbunătăți managementul resurselor și al aplicațiilor în sistemele distribuite. Caracterul puternic dinamic al mediilor Grid, în care atât resursele cât și potențialii utilizatori accesează sistemul cu o frecvență ridicată, definește cerințe complet noi pentru strategiile de monitorizare și face imposibilă aplicarea unor tehnologii și soluții cunoscute în sistemele "clasice", inclusiv în sistemele distribuite de dimensiuni mici sau medii. Pentru a obține o soluție viabilă de monitorizare în astfel de medii dinamice, în cadrul proiectului s-au utilizat informații provenite direct din sistemele de operare. O altă posibilitate de a îmbunătăți gestiunea resurselor în sisteme Grid dinamice este de a utiliza algoritmi de predicție pentru a estima starea resurselor în momentele imediat următoare.

Dezvoltarea unui instrument de simulare util în compararea algoritmilor de planificare. Eterogenitatea și autonomia site-urilor componente sunt aspecte de bază în mediile Grid. Datorită acestor aspecte, este dificil de experimentat și testat algoritmi de planificare, deoarece prezenta unui număr mare de utilizatori face aproape imposibilă repetarea unui experiment de mai multe ori în condiții identice. De asemenea, din cauza autonomiei site-urilor accesul la acestea poate fi limitat uneori, în momentele în care utilizatorii locali au aplicații importante de executat. Din aceste motive, un instrument de simulare este foarte util pentru a realiza teste inițiale pentru algoritmi de planificare. Instrumentul de simulare dezvoltat în cadrul acestui proiect oferă facilități diverse, cum sunt simularea execuției aplicațiilor, a transferurilor în rețea și a operațiilor cu baze de date. Extinderea instrumentului de simulare pentru testarea mai multor tipuri de algoritmi de planificare. Un avantaj important al instrumentului de simulare este flexibilitatea, putându-se modela diverși algoritmi de planificare prin simpla creare a unor noi clase Java care să descrie algoritmi, implementând o interfață specifică. Astfel au fost simulate și testate atât strategii clasice de planificare, cât și strategii noi propuse în cadrul proiectului (bazate pe algoritmi genetici).

Propunerea de noi soluții pentru planificarea în medii Grid. Problema planificării este în sine foarte dificilă. Costul găsirii unei soluții optime pentru planificare este în cele mai multe cazuri prohibitiv, motiv pentru care sunt de preferat euristicele care ajung la soluții apropiate de cea optimă. Un exemplu de euristica este cel propus în cadrul acestui proiect, bazat pe



UNIUNEA EUROPEANĂ



GUVERNUL ROMÂNIEI



Instrumente Structurale
2007 - 2013

algoritmi genetici. Testele efectuate au arătat ca algoritmi genetici ajung la soluții mai bune decât cei clasici, obținând un echilibru aproape de maxim în încărcarea resurselor din sistem. De asemenea, execuția acestor algoritmi se face cu costuri mai scăzute, iar implementarea este facilă. O altă abordare pe care o propunem este cea bazată pe modelul economic, având în vedere similitudinile dintre mediile Grid și sistemul economiei de piață. Ca și în acest sistem, în Grid accentul este pus pe calitatea serviciilor oferite consumatorilor în raport cu beneficiile obținute de ofertanții de servicii.

Arhitecturi orientate pe servicii. Conceptul de arhitectura orientată pe servicii a început să se contureze recent și a avut o evoluție rapidă, fiind adoptat în tot mai multe sisteme informatice în ultimii ani. Totuși, majoritatea ideilor care stau la baza arhitecturilor orientate pe servicii nu sunt noi: structurarea aplicațiilor în componente independente, care să funcționeze ca servicii, oferind anumite funcționalități, a fost introdusă cu peste 20 de ani în urmă prin programarea orientată pe obiecte. De asemenea, comunicarea între aplicații/servicii, pe baza unui anumit protocol, a fost implementată prin mecanisme ca DCOM și CORBA încă de când a început evoluția sistemelor distribuite.



Activitatea 3.1. Proiectarea aplicațiilor ce vor fi dezvoltate prin explorarea proiectelor similare la nivel european și internațional, pentru a asigura interoperabilitatea

Activitatea 3.1 are ca scop identificare modelelor științifice și proiectarea aplicațiilor ce vor fi dezvoltate. S-au analizat principiile, metodele, modelele și proiectele existente la nivel internațional și au fost identificate posibilitățile de utilizare. Astfel, vor fi descrise componentele de monitorizare, de simulate și planificare în sistemele distribuite. La finalul secțiunii vom prezenta modelul SOA pentru realizarea aplicațiilor.

Monitorizarea în sistemele distribuite

Soluții existente de monitorizare pentru sisteme distribuite. Mecanismul de monitorizare în sistemele Grid are ca scop formarea unei imagini cât mai clare despre ceea ce înseamnă evoluția în timp a sistemului. Informațiile de monitorizare sunt importante în procesele de decizie pentru partea de planificare, de analiză a performanțelor sistemelor și aplicațiilor. Vom descrie pe scurt modelele arhitecturale folosite în prezent pentru monitorizarea sistemelor distribuite pe scara largă. Aceste soluții se circumscriu unui model generic structurat pe trei niveluri ale monitorizării: nivelul inferior, al colectării datelor de monitorizare reprezentat de ansamblul senzorilor care furnizează informațiile de interes despre sistemele observate, nivelul intermediar, al gestiunii datelor colectate și nivelul superior, al prezentării informațiilor de monitorizare utilizatorilor interesați.

Pentru a realiza interoperabilitatea între diversele sisteme existente, Open Grid Forum a introdus un set de standarde și modele arhitecturale pentru diferitele aspecte ale mediilor Grid. Între acestea se găsește GMA (Grid Monitoring Architecture), o arhitectura de bază pentru monitorizare în Grid, bazată pe interacțiunea dintre trei componente: producători, consumatori și servicii de repertoriere. Sistemul de monitorizare R-GMA (Relational GMA) este o implementare a arhitecturii GMA care are două proprietăți specifice: nu este necesară cunoașterea serviciului de repertoriere (directory service) pentru a furniza sau obține informații (deoarece comunicarea cu serviciul este gestionată de producător și consumator) și întregul sistem funcționează și poate fi interogată ca o bază de date relațională. O altă soluție de monitorizare existentă care implementează GMA este GridICE, un proiect care a evoluat în cadrul EGEE. Acesta folosește o ierarhie pe două niveluri pentru colectarea datelor: colectare locală site-ului (cluster-ului) și colectare la nivel de Grid. GridICE furnizează mai multe moduri de accesare a informațiilor de monitorizare: o interfață web cu reprezentări textuale și grafice, o reprezentare XML care poate fi folosită de aplicații dar și un mecanism de publish / subscribe pentru notificări ale evenimentelor detectate.

Descrierea soluției de monitorizare propuse. Serviciul de Monitorizare a Grid-ului are scopul de a colecta informații privind starea Grid-ului, acesta fiind un mediu eterogen și dinamic. În acest sens soluția propusă se bazează pe sistemul distribuit de monitorizare MonALISA împreună cu ApMon, o aplicație de monitorizare care poate fi configurat pentru a trimite informații de stare în forma unor datagrame UDP către MonALISA. Sunt obținute astfel informații despre noduri și despre clustere, informații de rețea (trafic, conectivitate, topologie etc.) pentru WAN și LAN, informații de monitorizare a performanțelor aplicațiilor, joburilor și serviciilor. Informațiile de monitorizare despre nodurile din cluster sau Organizația Virtuală vor fi preluate de către Componenta de Planificare pentru a optimiza algoritmi folosiți în funcție de informațiile primite în timp real de la sistemul de monitorizare. Pentru colectarea informațiilor, se va implementa o variantă a Clientului de MonALISA, care se conectează la serviciul de monitorizare prin servere proxy și obține date pentru algoritmi de planificare folosiți. Astfel se introduc în calcule informații recente despre starea Grid-ului, pentru a obține timpi de execuție estimați cât mai aproape de cei reali.

Sistemul MonALISA este proiectat ca un ansamblu de subsisteme autonome, multi-thread, autodescriptive și bazate pe agenți care sunt înregistrate ca servicii dinamice, și sunt capabile să colaboreze și să coopereze în realizarea unei structuri masive de informație și sarcini de procesare. Acești agenți pot analiza și procesa informația, într-un mod distribuit, pentru a asigura optimizarea deciziilor în aplicații la o scară largă. O arhitectură bazată pe agenți oferă abilitatea de investire a sistemului cu grade crescătoare de inteligență, pentru a reduce complexitatea și a face sisteme globale administrabile în timp real. Scalabilitatea sistemului derivă din folosirea motorului de execuție multithreaded pentru găzduirea unei varietăți de servicii dinamice sau agenți și abilitatea fiecărui serviciu să se înregistreze la rândul său și apoi să fie descoperit și folosit de orice alte servicii, sau clienți care solicită asemenea informație. Sistemul este proiectat să integreze cu ușurință unele existente de monitorizare și procedurile și să ofere această informație într-un mod dinamic și obișnuit oricărui alt serviciu sau clienț. Scalabilitatea unui sistem derivă din uzul unui motor multi-threaded pentru a găzdui o varietate de servicii dinamice, abilitatea fiecărui serviciu să se înregistreze la rândul său și apoi să fie descoperit și folosit de orice alt serviciu sau clienț care solicită asemenea informație. Mediul integrează mai multe proceduri menite să colecteze parametrii ce descriu nodurile computaționale, aplicațiile și performanța rețelei. Agenții specializați mobili sunt folosiți în cadrul MonALISA pentru a realiza sarcinile de optimizare globală sau a ajuta și a îmbunătăți operația pentru sisteme distribuite mari prin realizarea unor sarcini de supraveghere pentru diferite aplicații sau parametrii în timp real. MonALISA rulează în prezent pe aproximativ 360 de site-uri.

Folosirea ApMon în cadrul soluției propuse permite mai departe particularizarea parametrilor monitorizați în funcție de aplicația țintă, extinzând astfel setul standard de parametrii (ex: load, memorie liberă disponibilă, spațiu pe disc) cu parametrii specifici fiecărei aplicații monitorizate. API-ul simplu pus la dispoziție permite integrarea facilă cu orice aplicație și injectarea transparentă a parametrilor în sistemul MonALISA. Pentru vizualizarea statisticilor globale, de istorie sau timp real cu parametrii monitorizați se va folosi clientul de tip Repository. Acesta permite stocarea optimizată a datelor colectate de la toate serviciile

distribuite, accesarea și vizualizarea lor, exportarea în alte aplicații folosind interfețe web, wap sau servicii web, agregarea datelor și automatizarea deciziilor în funcție de politicile specificate de utilizatori. Detaliem în secțiunile următoare cele 2 componente folosite pentru procesul de monitorizare Grid: ApMon și Repository-ul MonALISA.

ApMon este un set de biblioteci flexibile care pot fi utilizate de orice aplicație pentru a transmite informații de monitorizare serviciilor MonALISA. Datele de monitorizare sunt trimise sub forma unor datagrame UDP către una sau mai multe mașini pe care sunt active servicii MonALISA. Aplicațiile pot raporta periodic orice tip de informații pe care utilizatorul dorește să le colecteze sau să le monitorizeze, existând posibilitatea ca pe baza lor să fie activați agenți de decizie sau să fie declanșate alarme ce semnaleză condiții de eroare. Bibliotecile ApMon sunt disponibile pentru 5 limbaje de programare: C, C++, Java, Perl și Python; pot fi utilizate atât în programe complexe de procesare de date, cât și din scripturi sau programe utilitare. Principalele avantaje ale ApMon sunt: flexibilitatea, configurarea dinamică, performanța ridicată a comunicăției, informația este stocată într-un mod structurat în bazele de date MonALISA.



Figura 1. Scenariu de utilizare al aplicației ApMon

Procesul de instalare ApMon este simplu și nu necesită alte componente software, ceea ce face ca acest instrument să reprezinte o soluție rapidă și de cost scăzut pentru a monitoriza atât job-uri, cât și sisteme. Rutinele oferite de către biblioteca asigură codificarea datelor de utilizare în formatul XDR, precum și construirea și trimiterea de datagrame UDP. După cum se observă în Figura 1 aplicațiile pot utiliza API-ul ApMon pentru a trimite valori pentru parametri de diverse tipuri unuia sau mai multor servicii MonALISA. Utilizarea protocolului UDP aduce avantajul unei performanțe ridicate a comunicăției (între o aplicație instrumentată cu ApMon și un serviciu MonALISA se pot transmite sute și chiar mii de datagrame pe secunda). Pierderea ocazională a câtorva datagrame nu este fatală, deoarece scopul utilizării este de a obține informații statistice; în plus, trimiterea se face în mod uzual la intervale de timp suficient de mici pentru ca pierderea unei datagrame să nu fie critică.

Adresele serviciilor MonALISA către care se trimit date de monitorizare cu ApMon pot fi specificate fie în codul sursa al aplicației, fie în fișiere locale sau la distanță, accesibile prin HTTP. ApMon se poate auto-reconfigura în mod dinamic, reincărcând periodic fișierele și paginile web de configurare. Pentru a evita problemele de securitate, serviciile MonALISA pot stabili o politică de a controla acceptarea mesajelor ApMon. Există două posibilități de a stabili care datagrame ApMon sunt acceptate: fie prin stabilirea unei parole ce trebuie să fie inclusă în datagramă, fie prin liste de adrese IP care sunt acceptate sau respinse. Aceste două metode pot fi utilizate simultan.

Monitorizarea automată a parametrilor de sistem și a job-urilor. Începând cu versiunea 2.0, ApMon oferă posibilitatea de a transmite, dintr-un fir de execuție separat, datagrame suplimentare ce conțin informații de monitorizare a sistemului, a aplicației care utilizează ApMon și chiar și a altor aplicații. Datagramele pentru monitorizarea sistemului includ valori curente pentru parametri cum ar fi load-ul procesorului, numărul total de procese, cantitatea de memorie liberă, spațiul disponibil pe disc etc. Alt tip de parametri de sistem sunt cei pentru care valorile transmise sunt mediate pe intervalul de timp din momentul transmiterii datagramei precedente până la momentul curent. Astfel de parametri sunt procentajul de timp în care procesorul a executat instrucțiuni în modul utilizator sau în modul sistem (raportat la timpul total), sau numărul mediu de KB transferați pe secundă prin fiecare interfață de rețea. Există și parametri de sistem statici, cum ar fi numele sistemului, cantitatea totală de memorie, adresele IP sau frecvența procesorului. Datagramele pentru monitorizarea job-urilor conțin valori pentru parametri cum ar fi cantitatea de memorie, spațiul pe disc și timpul de procesor consumate. Se pot monitoriza job-uri multi-proces, în acest caz făcându-se o sumă a cantităților de resurse consumate pentru toate subprocesele

Repository MonALISA. Mediul de monitorizare MonALISA oferă un mecanism simplu de a crea clienți capabili să folosească mecanismul de descoperire din JINI și să găsească toate serviciile care rulează. Acești clienți se pot înregistra la un set de parametri sau agenți de filtrare pentru a primi anumite informații de la toate serviciile. Aceasta oferă posibilitatea de a prezenta vizualizări globale ale ansamblului dinamic de servicii care rulează într-un mediu distribuit, pentru a fi folosite de serviciile de nivel superior. Valorile primite sunt stocate local într-o bază de date relațională (MySQL sau Postgres), asupra lor aplicându-se proceduri dedicate de compresie pentru datele mai vechi.

Informația astfel stocată este folosită pentru a crea un repository web, capabil să ofere o viziune sintetică asupra modului de comportare al sistemelor distribuite la scară mare. Un mecanism bazat pe servlet e folosit pentru a prezenta grafice de timp real, de istorie, statistici, tabele de o manieră flexibilă. Odată pornit, repository-ul se înregistrează cu anumite predicte și stochează valorile primite într-o bază de date. Un predicat are următoarea formă: Ferma / Cluster / Nod / timp_start / timp_final / lista de funcții. Acești parametri sunt ulterior afișați folosind mecanismul de servlet într-o largă varietate de grafice, tabele statistice, bazate pe fișierele de configurare care descriu vizualizările dorite și parametrii proprii acestora, oferind astfel perspective globale sau particulare.

Repository-ul folosește de asemenea o hartă interactivă pentru vizualizarea poziționării fermelor dar și pentru monitorizarea anumitor servicii, legăturilor sau traficului. Astfel este oferită funcționalitatea clientului interactiv disponibil în MonALISA într-o interfață adaptată web, specializată în interpretarea datelor. Figura 2 prezintă o perspectivă schematică asupra interacțiunii dintre aceste servicii:

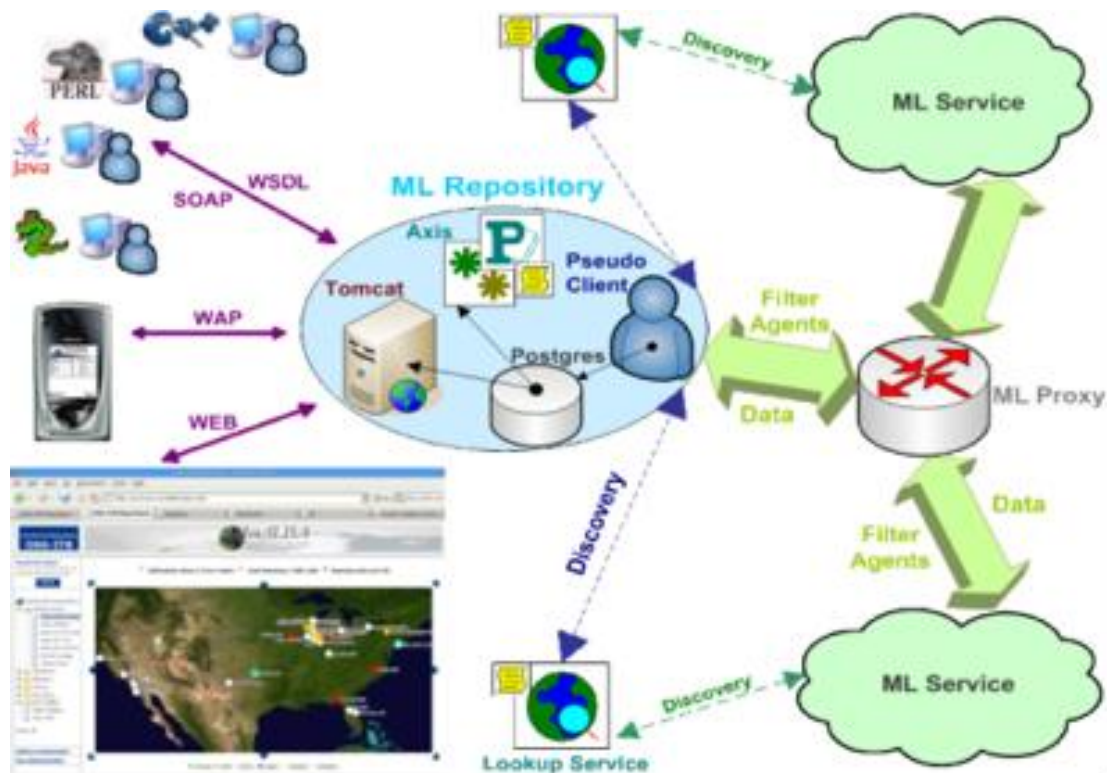


Figura 2. Arhitectura Repository-ului MonALISA

Bazat pe datele stocate în baza de date, și pe fișierele de configurare se pot construi tabele statistice sau grafice de diverse tipuri: barchart, serii, spider, pie, axe multiple, tabele, serii cu evenimente asociate etc. Aceste grafice plotează parametrii monitorizați: încărcarea siturilor, număr de CPU, număr de joburi, tipuri de joburi, joburi idle, joburi running, joburi pe VO, utilizarea nodurilor, parametrii sistem (memorie, spațiu disc, etc), trafic WAN, lățime de bandă disponibilă, pathload, statusul siturilor monitorizate, etc. oferind o viziune globală asupra sistemului distribuit pentru a putea lua decizii de optimizare, sau o integra într-un sistem de triggerare bazat pe anumite evenimente. Asupra datelor se pot adăuga diverse niveluri de agregare, selectabile de către utilizatori din pagina de web: sumare, mediere, integrare, scări logaritmice pentru o mai bună înțelegere a informației afișate. Utilizatorul dispune de asemenea și de opțiuni asupra modului de afișare: selecția tipului de grafic, a perioadei de plotare, selecția fermelor, a statisticilor generate pe baza graficului selectat, etc.

Fiind folosit pentru stocarea și interpretarea datelor de la toate serviciile pe perioade mari, baza de date ajunge rapid la proporții mari iar query-urile adresate ei prin mecanismul de servlet pot deveni astfel consumatoare de timp. Pentru a rezolva această problemă se

folosesc mai multe abordari: tabele de stocare a datelor cu granularitari diferite (pentru cererile pe o perioada mare de timp – ex: luni/ani - sunt folosite tabele cu o rezolutie de timp mai mare – ex: 15 min iar pentru cele pentru o perioada mai scurta – ora / zi / saptamana – tabelele cu o rezolutie de timp mica -ex: 1 min); pentru a oferi rapid informatiile în timp real, ultimele valori primite de la servicii sunt stocate intr-un buffer de memorie cu o dimensiune adaptabila (fereastra de timp acoperita difera în functie de memoria disponibila) de unde sunt rapid servite servletilor, înainte de a fi scrise în baza de date; exista un cache indexat dupa cele mai frecvente cerei din servleti, cu o fereastra de timp selectabila, iar ploturile sunt servite intai din acest cache, cererea pe baza de date fiind facuta doar în absenta rezultatului din cache; în baza de date sunt stocate tabele cu indexari multiple dupa parametrii / serii pentru a permite un raspuns mai rapid.

Stocarea intr-un loc centralizat (eventual replicat) a informatiilor globale despre sistemul distribuit monitorizat permite astfel aplicarea unor algoritmi de detectie de pattern-uri, stabilire de corelatii între evenimentele monitorizate și folosirea acestor informatii pentru automatizarea procesului de gestiune a resurselor și de administrarea a Organizatiilor Virtuale.

Simularea sistemelor distribuite

Sistemele distribuite de mari dimensiuni ce presupun agregarea și partajarea resurselor folosind rețele globale prezintă noi provocări pentru oamenii de știință. În cadrul proiectului MONARC au fost propuse soluții la problema studierii științifice și sistematice a tehnologiilor distribuite prin folosirea modelării ca instrument pentru testarea de nivel înalt a aplicațiilor, resurselor și rețelelor specifice acestor sisteme. Modelul de simulare original dezvoltat în contextul simulatorului MONARC este generic, incorporând componentele și mecanismele necesare creării și execuției cu succes a unei game largi de experimente realiste de simulare a unor diverse arhitecturi de sisteme distribuite de mari dimensiuni, cuprinzând resurse și tehnologii ce pot varia de la transferuri de date la planificare și replicări de date, în care resursele conlucrează împreună pentru furnizarea unui set comun de caracteristici.

MONARC înglobează astfel o gamă largă de algoritmi și tehnologii de ultimă oră, permițând simularea realistă a unei game largi de tehnologii și sisteme distribuite, în conformitate cu componentele și caracteristicile specifice acestora. Maturitatea simulatorului este demonstrată de numărul mare de scenarii executate cu succes

Prin comparație cu alte instrumente de simulare ce se adresează sistemelor distribuite, soluția prezentată în cadrul acestei lucrări oferă o serie de avantaje. Una dintre problemele cu care se confruntă astfel de aplicații ține de scepticismul utilizatorilor în legătură cu validitatea rezultatelor obținute. MONARC a fost testat cu succes în diverse experimente ce au urmărit atât verificarea validității modelului folosit și a soluțiilor de implementare, cât și verificarea performanțelor oferite de instrumentul de simulare.

De asemenea majoritatea instrumentelor de simulare existente sunt focusate pe anumite aspecte ale sistemelor distribuite de mari dimensiuni (precum replicarea datelor sau planificarea execuției proceselor) și nu permit modelarea decât a unui set restrâns de posibile scenarii. MONARC este un instrument generic ce poate fi folosit cu succes pentru simularea unei game largi de posibile scenarii, de la modelarea de protocoale de rețea până la sisteme distribuite de mari dimensiuni cuprinzând o varietate mare de resurse sau aplicații din cele mai diverse domenii.

Simulatorul pentru sisteme distribuite MONARC. MONARC furnizează un mediu de simulare realist pentru modelarea sistemelor de calcul distribuite de mare întindere, oferind flexibilitate și dinamism în evaluarea performanțelor unei game largi de arhitecturi de procesare. Simulatorul conține mecanismele necesare modelării traficului concurrent și evaluării de diverse strategii de replicare a datelor sau de planificare a execuției sarcinilor de lucru. Caracteristicile aplicației precum arhitectura robustă, implementarea modulară și extensibilă, setul bogat de componente disponibile, fac din MONARC un instrument de modular deosebit de popular în rândul utilizatorilor din diverse domenii științifice. Folosirea tehnologiilor de monitorizare în procesele de validare și proiectare au consolidat și mai mult succesul de care se bucură acest proiect.

Componentele și arhitectura simulatorului. Arhitectura aplicației de simulare este special proiectată pentru modelarea unor sisteme distribuite de mari dimensiuni, cuprinzând interacțiuni complexe și schimbul și procesarea unor cantități impresionante de date. Cea mai potrivită soluție pentru modelarea unor astfel de sisteme a reprezentat-o adoptarea unei proiectări bazate pe obiecte. O astfel de abordare permite, printre altele, modelarea sistemelor a căror stări suportă modificări dinamice în timp, maparea directă și facilă a componentelor logice peste programul de simulare și furnizarea mecanismelor de interacțiune necesare modelării unor sisteme distribuite de mare întindere (vezi Figura 3).

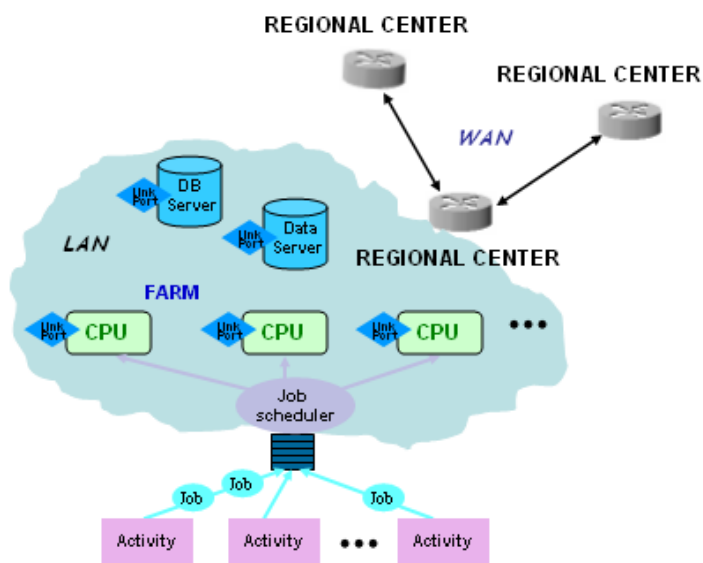


Figura 3. Componentele modelului de simulare

Simulatorul MONARC folosește o abordare orientată spre procese pentru implementarea simulării bazate pe evenimente discrete, aspect ce facilitează simularea aplicațiilor concurente și a modelelor stocastice. Obiectele ce folosesc fire de execuție, numite și „Obiecte Active” (ce folosesc fire de execuție pentru implementarea secvenței de rulare și care încorporează un contor de program), permit o mapare naturală a comportamentului specific procesării distribuite de date în simulator. Modelarea procesării concurente se realizează pe baza mecanismului de „întreruperi” ce este implementat la nivelul motorului de simulare. Simulatorul MONARC este implementat folosind tehnologia Java, beneficiind de suportul adecvat oferit de aceasta pentru dezvoltarea unor simulărilor dinamice și distribuite orientate spre procese. Java include suport nativ pentru proiectarea multi-thread a procesării concurente, aspect ce la nivelul simulării a fost folosit în combinație cu implementarea unui mecanism de planificare dedicat. Motorul de simulare beneficiază astfel de proprietăți ce îi permit instalarea ușoară și rularea unor scenarii complexe chiar și pe sisteme multi-procesor. O contribuție importantă la dezvoltarea proiectului a fost adusă de numeroșii pași de evaluare și de optimizare execuțai ce au dus la creșterea performanței deosebite a motorului de simulare.

Motorul de simulare. Nivelul de jos al arhitecturii aplicației de simulare este reprezentat de motorul de simulare. Acesta este responsabil cu gestionarea firelor de execuție, a evenimentelor de simulare, a cozilor de evenimente și cu gestionarea mecanismului prin care task-urile interacționează și implementează algoritmul de simulare bazată pe evenimente discrete (DES).

Motorul de simulare furnizează un mecanism dedicat bazat pe semafoare pentru gestiunea obiectelor active (sau task-uri). Acest mecanism folosește suportul nativ, superior în termeni de performanță, al tehnologiei Java pentru procesarea concurentă. Tehnologia Java furnizează câteva avantaje: suport nativ pentru multithreading, paradigma programării orientate pe obiecte și portabilitate. De altfel portabilitatea permite folosirea aplicației de simulare pe o mare varietate de platforme (sisteme mono sau multi-procesor rulând Linux, Solaris sau Windows).

Task-urile de simulare pot gestiona execuția activităților de simulare sau pot reprezenta componente ale sistemelor simulate. Un task de simulare reprezintă o entitate ale cărei proprietăți se modifică dinamic în timp, comportamentul acestuia fiind influențat direct de evenimentele de simulare. În cadrul simulatorului task-urile interacționează prin schimburi de evenimente. Execuția acestora este gestionată de către un planificator centralizat dedicat. Acest planificator are rolul de a gestiona atât execuția task-urilor, cât și a evenimentelor de simulare produse.

Componente de simulare. Componentele furnizate de MONARC sunt folosite pentru modelarea atât a resurselor sisteme distribuite cât și a comportamentelor aplicațiilor ce rulează pe astfel de sisteme. Pentru a furniza un cadrul generic, modelul de simulare conține

componente ce modelează comportamentul diverselor resurse de procesare, de date, de rețea și pe cele ale diverselor activități corespunzătoare aplicațiilor distribuite.

Arhitectura aplicației de simulare este bazată pe modelul Centrului Regional. Centrul regional reprezintă un obiect complex compozit ce conține servere de date și noduri de procesare, toate conectate prin intermediul unei rețele locale. Un centru regional poate fi conectat cu alte centre prin intermediul ruterelor și al rețelelor de mare întindere. De asemenea un centru regional poate instanția dinamic un număr de obiecte ce modelează utilizatori și activitățile acestora. Într-un centru regional maparea sarcinilor de lucru pe nodurile de procesare se realizează conform unor scenarii de planificare.

Activitatea reprezintă utilizatorii sistemului modelat. Aceștia pot genera sarcini de lucru de procesare de date pe baza unor scenarii. Fiecare centru regional poate avea una sau mai multe astfel de activități ce submit sarcini de lucru. Entitatea farm gestionează nodurile de procesare ale centrului regional. Rolul acesteia este de a aștepta submiterea de sarcini de lucru generate de activități și de a apela planificatorul adecvat pentru execuția acestora. Obiectul CPUUnit descrie comportamentul unei stații de lucru aparținând sistemului modelat. Acesta este caracterizat printr-o cantitate de putere de procesare și o cantitate de memorie. Modelarea ocupării memoriei poate include un mecanism de paginare, aspect ce asigură modelarea corectă a comportamentului specific diverselor sisteme de operare. Cantitatea de putere de procesare este alocată sarcinilor de procesare în funcție de un mecanism de prioritate. Utilizatorul poate astfel specifica sarcini având priorități mai ridicate decât altele pentru execuție.

O extensie a unității de procesare este reprezentată de CPUCluster. Acest obiect modelează un cluster de unități de procesare. În afară de puterea de procesare și cantitatea de memorie disponibilă, el este caracterizat și printr-o cantitate de spațiu de stocare. Această clasă include mecanismele modelării inclusiv a defectării diverselor unități de procesare incluse. CPUCluster este adecvat pentru modelarea mediilor Grid de mare complexitate, în timp ce CPUUnit este mai adecvat modelării sistemelor distribuite constând într-un număr redus de noduri.

Planificarea în sisteme distribuite

Planificarea în sistemele Grid este folosită pentru a “gestiona eficient și într-o manieră eficientă accesul la folosirea resurselor Grid de către utilizatori” [13]. Conform [14] planificarea în sistemele Grid este compusă din trei componente principale: Consumatorii, Resursele și Politica (vezi Figura 4).

Cele două comunități majore implicate în Grid, consumatorii de resurse care submit diverse aplicații, respectiv furnizorii de resurse care își partajează resursele, au de obicei motivații diferite când intra în Grid. Aceste motivații sunt reunite în funcțiile obiectiv. În prezent, majoritatea funcțiilor obiectiv în Grid sunt mostenite din sistemele distribuite și paralele tradiționale. Utilizatorii Grid sunt în principal interesați de performanța propriilor aplicații, spre

exemplu costul total pentru a rula o anumita aplicatie, în timp ce furnizorii de resurse sunt de obicei mai atenti la performanta propriilor resurse, de exemplu utilizarea lor în anumite perioade. Astfel functiile obiectiv pot fi clasificate în doua categorii: centrate pe aplicatii si centrate pe resurse.

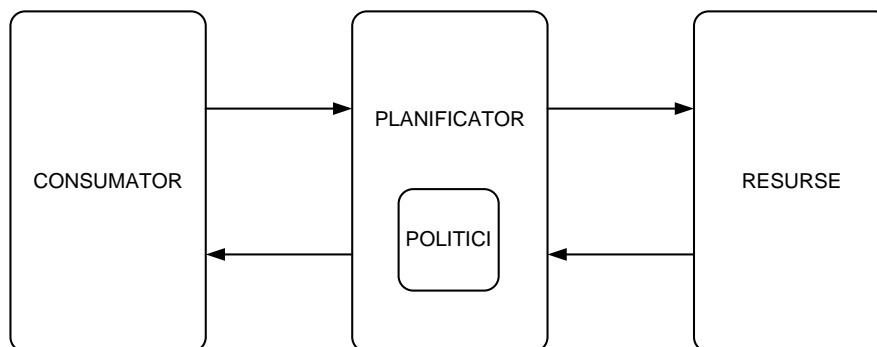


Figura 4. Modelul general de planificare

Obiective de planificare centrate pe aplicații. Acești algoritmi încearcă să optimizeze performanța fiecărei aplicații. Majoritatea obiectivelor aplicațiilor Grid curente sunt legate de timp, spre exemplu makespan, timpul trecut de la începutul primului task dintr-un job până la sfârșitul ultimului task din job. Pe măsură ce modelele economice sunt introduse în Grid, costul economic al unei aplicații devine o preocupare pentru utilizatori, respectiv costul plătit de o aplicație pentru a folosi o anumită resursă. Pe lângă aceste funcții simple multe aplicații folosesc funcții compuse, spre exemplu unele doresc atât timp redus de execuție și costuri economice mici. Dificultatea principală în adoptarea unor astfel de funcții stă în normalizarea celor două metrici diferite: timpul și banii. Odată cu dezvoltarea infrastructurii Grid s-a observat o tendință orientată spre servicii, astfel încât calitatea serviciului (QoS) capătă o importanță marită. Sensul QoS este dependent de aplicații particulare, de la capacități hardware la software. De obicei, QoS este o constrângere impusă proesului de planificare în locul funcției obiectiv finale.

Obiective de planificare centrate pe resurse. Aceste obiective sunt legate de utilizarea resurselor, spre exemplu throughput – capacitatea unei resurse de a procesa un anumit număr de joburi într-o perioadă anumită de timp; utilizare – procentajul de timp în care o resursă este ocupată. O utilizare scăzută înseamnă că resursa este idle și astfel irosită. Pentru o resursă multiprocesor, utilizări diferite între procesoare descrie de asemenea echilibrul încălzirii sistemului și o scădere a throughput-ului. Condor [Condor07] este un sistem care adoptă throughput-ul ca politică de planificare. O altă politică viabilă de administrare a resurselor o constituie profitul economic – beneficiul economic adus de atragerea aplicațiilor utilizatorilor Grid pentru a rula pe propriile resurse.

Funcțiile obiectiv menționate anterior dau rezultate bune făcând o presupunere, anume că resursele sunt dedicate deci furnizează performanță constantă unei aplicații. În această presupunere nu ține într-un mediu Grid așa cum am arătat în secțiunea precedentă. Spre exemplu, să presupunem că a fost găsită o planificare optimă cu makespan OPT, dacă

resursele implicate sunt stabile. Dacă resursele Grid sunt brusc încetinite (interupte de joburi ale utilizatorilor locali, legătura la rețea, etc.) și această situație persistă pentru o perioadă mai mare de timp, atunci makespan-ul planificării reale este departe de OPT și nu poate fi limitat de algoritmi de planificare care nu pot prevedea schimbările în performanță. Deci, dacă funcția obiectivă a unei planificări este makespan-ul și nu există o limită fixă pentru performanțele resurselor fie pentru perioada de timp a execuției, cu alte cuvinte, dacă nu putem prevedea fluctuațiile de performanță, nu va exista în general un algoritm de aproximare makespan care să fie aplicabil în Grid.

Pentru a elimina acest neajuns, a fost propus un nou criteriu de planificare: consum total de cicluri procesor (CTCP) – numărul total de instrucțiuni pe care Gridul le poate executa de la timpul de start al planificării până la terminare. CTCP reprezintă puterea de calcul totală consumată de o aplicație. Cu acest nou criteriu, lungimea unui task este numărul de instrucțiuni din task; viteza unui procesor este numărul de instrucțiuni executate în unitatea de timp; iar procesoarele dintr-un grid sunt eterogene deci au viteze diferite. În plus, viteza fiecărui procesor variază în timp datorită cursei între procese în cadrul unui sistem deschis. Avantajul CTCP este că este puțin afectat de variația performanțelor resurselor dar totuși legat de makespan.

Structura sistemului de planificare dezvoltat în UPB este prezentată în Figura 5. Ținând cont de sistemul distribuit real în care va funcționa planificatorul și datorită interacțiunii acestuia cu diferite alte aplicații (un exemplu ar fi folosirea datelor de monitorizare) arhitectura construită este formată din mai multe servicii. Componentele arhitecturii au ca rol descoperirea resurselor disponibile, rularea algoritmului de planificare, execuția și monitorizarea evoluției task-urilor în sistemul distribuit.

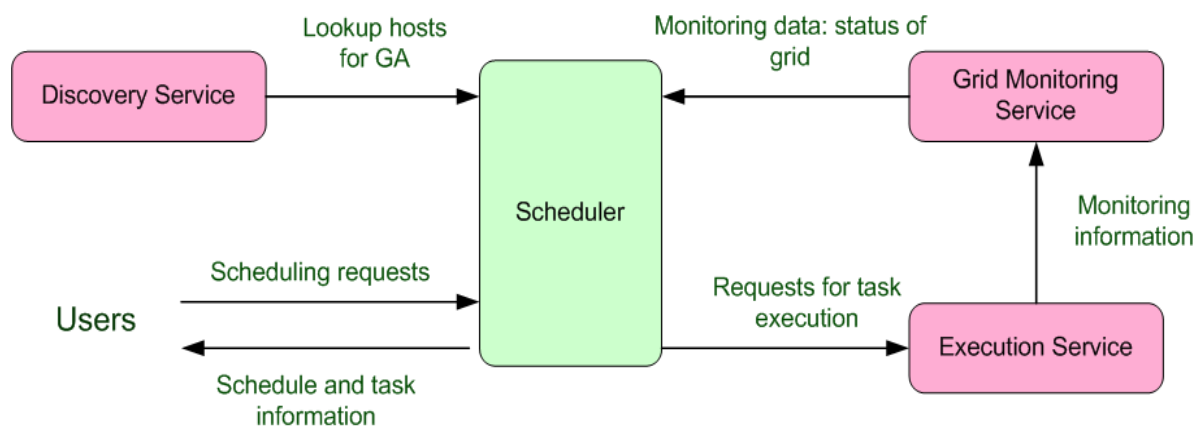


Figura 5. Structura sistemului de planificare DIOPGENES

Componentele funcționale ale sistemului, componente care interacționează cu planificatorul sunt: Serviciul de Cereri; Serviciul de Monitorizare a Grid-ului; Serviciul de Execuție; Serviciul de Look-up.

Algoritmul de planificare – partea centrală a întregului sistem – a fost dezvoltat pe baza unor modele din literatura de specialitate privitoare la planificatoare bazate pe algoritmi genetici. În construcția modelului, sarcinile trebuie să respecte restricții referitoare la timpii de execuție, la cerințele de viteză de procesare și de memorie pentru sistemul pe care o aceasta va rula, ținându-se totodată cont și de ordonarea în timp a sarcinilor.

În cadrul planificatorului DIOGENES un scop important este îmbunătățirea modului de descoperire a agenților. Deoarece este un planificator descentralizat acesta poate avea un rol foarte important pentru performanța și viteza cu care noii agenți sunt integrați și pot primi cereri de planificare.

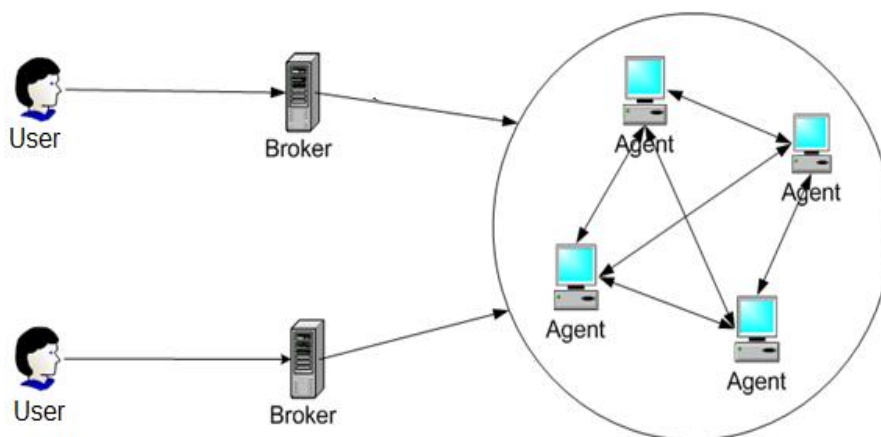


Figura 6. Modelul de comunicație în DIOGENES

Modelul actual al planificatorului, prezentat în Figura 6, se bazează pe tehnologia Jini [11]. Se folosește astfel un serviciu central numit lookup unde se înregistrează toți agenții. Acest serviciu asigură și comunicația între agenți funcționalitatea întregului sistem. Noii agenți care intră în sistem trebuie să se înregistreze printr-un protocol de descoperire la serviciul de lookup. După acest pas agentul primește informații despre interfețele implementate și despre moduri de comunicare cu alți agenți din sistem. Acesta este o metodă nesigură deoarece serviciul de lookup este centralizat și toți agenții și-ar pierde funcționalitatea în cazul în care acesta se defectează. În plus, un serviciu centralizat mai prezintă și probleme de scalabilitate în cazul în care numărul de agenți crește foarte mult.

O soluție posibilă ar fi folosirea unei tehnologii bazate pe JXTA [12]. Este o abordare superioară deoarece JXTA se bazează pe trimiterea de mesaje într-un timp optim în cadrul unor rețele punct-la-punct de mari dimensiuni. JXTA oferă de asemenea un limbaj abstract pentru modelarea comunicației între agenți. Se folosesc de asemenea mesaje în format XML pentru a putea fi ușor de împachetat, trimis și pentru o compatibilitate mai bună cu diferite medii de transport. Ca un avantaj față de Jini, care folosește RMI și serializarea obiectelor

Java pentru trimiterea de mesaje, formatul XML prezent în JXTA oferă o modularitate și o independență față de limbajul de programare folosit.

Marele avantaj al acestei tehnologii este faptul că nu folosește un server centralizat pentru managementul agenților care rulează la un moment dat, cum este cazul sistemului Jini. Se permite astfel folosirea oricărui agent existent deja pentru a înregistra noii veniți în sistem și pentru a răspândi informații corecte despre toate serviciile disponibile. Se elimină astfel problemele de scalabilitate întâlnite dacă ar fi fost nevoie interogarea unui server de fiecare dată când un agent dorea să cunoască starea altui agent din sistem. Se folosește în acest scop serviciul de descoperire JXTA care este complet descentralizat, prin folosirea de noduri speciale pentru acest scop, numite noduri “rendezvous” care intermediază procesul de descoperire.

Toți agenții rulează astfel independent, având posibilitatea să trimită mesaje de la unul la altul fără restricții din partea infrastructurii și la pornire și înregistrare va cunoaște în timp real starea oricărui alt agent din sistem. Comunicația se realizează prin canale de comunicație create special care se comportă ca linii seriale, indiferent de mediul de comunicație de la nivelele inferioare.

În plus, aceste avantaje ușurează foarte mult modul de programare a aplicațiilor distribuite. JXTA oferă o abstractizare a arhitecturii sistemului distribuit în cadrul căreia rulează aplicația și simplifică pașii necesari pentru coordonarea unui sistem de agenți, multe din mecanismele necesare făcându-se automat, fără intervenția dezvoltatorului.

Servicii Grid (implementarea SOA pentru Grid)

Arhitectura orientată pe servicii(SOA) se bazează pe conceptul de cuplare slabă. SOA definește un mod de dezvoltare a aplicațiilor care se bazează pe împărțirea în servicii individuale, în loc să fie focalizate pe aplicații formate din mai multe funcții [10].

Arhitectura orientată pe servicii în sistemele Grid. Avantajele arhitecturii orientate pe servicii au determinat adoptarea acestora și în sistemele Grid, într-un moment în care acest concept era încă nou. Orientarea pe servicii a fost o evoluție față de primul model arhitectural al Grid-ului, propus de Ian Foster, Carl Kesselman și Steven Tuecke în [15]. Acest model implica existența mai multor niveluri, aranjate în mod asemănător unei clepsidre: partea îngustă a clepsidrei corespunde unui set mic de protocoale și abstracții de bază; deasupra acestora pot fi mapate o multitudine largă de comportamente, iar dedesubt pot exista tehnologii de nivel jos foarte diverse.

De la acest model bazat pe protocoale s-a făcut trecere la unul orientat pe servicii. Astfel, [16] descrie Grid-ul ca fiind “un set extensibil de servicii care răspund la mesaje specifice protocoalelor”. Serviciile Grid pot fi agregate pentru a îndeplini cerințele organizațiilor virtuale. Acest model nou, denumit “Arhitectura Deschisă pentru Servicii Grid” (Open Grid Services

Architecture – OGSA), si-a propus sa alinieze tehnologiile Grid cu standardele serviciilor web. Astfel, se pot valorifica avantajele care rezulta din utilizarea limbajului WSDL (Web Services Description Language) si din separarea descrierii abstracte a interfeței pe care serviciul o ofera utilizatorilor, de specificarea concreta a adresei unde serviciul este disponibil si a modului în care acesta va comunica cu clientii. Printre aceste avantaje mentionam generarea automata a codului pentru client si pentru server pornind de la descrierea WSDL, posibilitatea de a descoperi servicii prin intermediul cataloagelor publice, asocierea descrierii serviciilor cu protocoale de retea interoperabile, compatibilitatea cu servicii de nivel mai înalt [16]. Modelul OGSA a a fost implementat în Globus Toolkit versiunea 3.

OGSA descrie mecanisme standard pentru crearea, denumirea si descoperirea instantelor de servicii Grid. În acest model, resursele computationale, legaturile de retea, resursele de stocare, bazele de date etc. sunt reprezentate de servicii; un serviciu poate fi definit ca o entitate care ofera anumite capabilitati, comunicand cu clientii prin intermediul rețelei. Perspectiva orientata pe servicii simplifica virtualizarea (incapsularea diverselor implementari în spatele unei interfețe comune) si raspunde la nevoia de a avea mecanisme standard de definire a interfețelor.

In 2003, [17] introduce un prim set de specificatii pentru conceptele OGSA, denumit “Infrastructura Deschisa pentru Servicii Grid” (Open Grid Service Infrastructure – OGSi). Autorii au definit modalitati pentru crearea, denumirea si gestionarea instantelor de servicii, pentru declararea si inspectarea datelor de stare ale serviciilor, pentru notificarea în cazul schimbarii starii etc. Definitiiile sunt date ca tipuri WSDL, si pot fi combinate pentru a crea servicii Grid complexe.

Versiunea urmatoare a Globus Toolkit (GT4) are în continuare la baza arhitectura orientata pe servicii, dar a adoptat o paradigma noua pentru dezvoltarea serviciilor Grid: Web Services Resource Framework (WSRF). Noua abordare ofera o compatibilitate mai buna cu instrumentele existente pentru dezvoltarea serviciilor web, inclusiv cele comerciale.

Globus Toolkit nu este singurul middleware pentru Grid în care s-a adoptat arhitectura orientata pe servicii. Un alt exemplu este Condor, un instrument pentru planificare de aplicatii si pentru gestiunea resurselor utilizat pe scara larga de mai mult de 20 de ani. La Condor a fost adaugata recent un modul numit BirdBath, care expune o parte din functionalitatile instrumentului ca servicii web. Arhitectura orientata pe servicii a fost introdusa si în instrumentele de planificare comerciale, cum ar fi Tivoli Dynamic Workload Broker de la IBM.

Tehnologiile Grid tind să folosească serviciile web pentru a se încadra în specificatiile SOA. Grid-ul, fiind un sistem distribuit este perfect pentru natura distribuită a SOA, permițând integrare ușoară, flexibilitate și acces sigur. Pot apărea totuși probleme de latență și concurență. În cadrul SOA tehnologiile de obicei includ SOAP, XML și standardele serviciilor web. XML formează partea de bază a tuturor standardelor, de la protocolul SOAP care este folosit pentru schimbul de informații până la metodele de descriere a serviciilor. De exemplu

WSDL este un limbaj de descriere bazat pe XML care este folosit pentru a descrie serviciile disponibile pentru clienții potențiali.

Mai există și un număr de probleme de securitate legate de folosirea arhitecturii orientate pe servicii în medii Grid. Există standarde de securitate pentru SOA care ar putea fi folosite, cum ar fi WS-Security, care oferă mecanisme de securitate de nivel scăzut cum ar fi integritatea mesajelor sau confidențialitatea, sau WS-Trust, care oferă și alte primitive foarte importante și extensii pentru schimbul de tokenuri de securitate și permite emiterea și diseminarea de credențiale în domenii de încredere diferite.

În articolul [8] se introduce o metodă de a insera rutine de securitate în jurul invocărilor de servicii în medii Grid și reprezintă una din soluțiile posibile pentru problema securității serviciilor. Acestea se folosesc pentru a securiza alocarea de resurse folosind mecanismele prezentate mai sus.

În exemplul din Figura 7 se poate observa o modalitate de inserare a unor proprietăți de securitate pentru invocarea unui serviciu web.

```
<gridInvoke ...>
  <security
    method="GSITransport |
      GSISecureMessage |
      GSISecureConversation"
    level="privacy | integrity"
    authz="none | self | host | anyString"?
    peer-credentials="filename"?
    anonymous="true | false"?
    delegation="none | full | limited"?
  />?
</gridInvoke>
```

Figura 7. Proprietăți de securitate pentru invocarea unui serviciu web

În mod normal când un client invocă un serviciu web sau trimite spre execuție un flux acesta nu se află pe aceeași mașină de pe care rulează clientul. O modalitate de configurare ar fi instalarea serviciilor disponibile pe anumite noduri în Grid unde clientul ar avea acces. La fel se poate proceda și pentru instalarea unui motor de fluxuri de servicii, cum ar fi BPEL. Clientul care invocă un serviciu va fi de fapt un portal sau un client SOAP simplu instalat pe mașina clientului.

Având în vedere faptul că este nevoie de un certificat Proxy al utilizatorului pentru a securiza apelurile prin criptare, aceste trebuie să fie disponibil în sistem. O soluție foarte simplă ar fi stocarea credențialelor utilizatorilor pe aceeași mașină pe care rulează portalul și să se creeze certificatele Proxy atunci când este nevoie. Aceasta este o metoda nesigura



UNIUNEA EUROPEANĂ



GUVERNUL ROMÂNIEI



Instrumente Structurale
2007 - 2013

deoarece necesită faptul ca o cheie privată a utilizatorului va fi stocată într-un loc de unde un atacator ar putea avea acces la ea. O soluție mai sigură ar fi adăugarea de informații despre credențialele utilizatorilor în headerele cererilor clienților de tip SOAP. Această metodă este mai sigură deoarece doar clientul este deținătorul cheii sale publice dar implică modificarea modului de invocare a serviciilor pentru a putea procesa și aceste informații adiționale.

În ambele cazuri se folosește serviciul MyProxy [9] de management al credențialelor. Acesta este o aplicație folosită pentru managementul certificatelor de tip X.509 care permite generarea de certificate Proxy pe baza credențialelor stocate ale utilizatorilor printr-o conexiune sigură de tip TLS. Utilizatorul va avea totuși nevoie să creeze direct acest certificat Proxy care are o durată de viață scurtă. Deoarece timpul de rulare a unui serviciu invocat este necunoscut se vor crea și mecanisme de reînnoire a certificatelor dacă acesta depășește durata de viață a certificatului Proxy.

Soluția prezentată oferă un grad înalt de securitate deoarece folosește doar conexiuni sigure pentru transferul de date, cum ar fi TLS pentru generarea certificatelor Proxy prin MyProxy sau HTTPS pentru invocările directe. De asemenea, este o soluție care se poate integra în middleware-ul Globus Toolkit care oferă managementul sistemului Grid.



Bibliografie

- [1] Dobre, C. M., C. Stratan, "MONARC Simulation Framework", RoEduNet International Conference, Timisoara, Romania, 2004.
- [2] Legrand, I. C., C. Cirstoiu, S. McKee, H. Newman, R. Voicu, C. M. Dobre, "VINCI : Virtual Intelligent Networks for Computing Infrastructures", în Proc. of Computing for High Energy Physics, Mumbai, India, February 13-17, 2006.
- [3] Legrand, I. C., C. M. Dobre, C. Stratan, "MONARC 2 – Distributed Systems Simulation", Technical Report, Switzerland, 2003.
- [4] Lin, H., "Economy-Based Data Replication Broker Policies în Data Grids", Bachelor of Computer Science paper, 2005.
- [5] Lin, H., J. H. Abawajy, R. Buyya, "Economy-Based Data Replication Broker", eScience, 2006.
- [6] Pop, F. C. M. Dobre, G. Godza, V. Cristea, "A Simulation Model for Grid Scheduling Analysis and Optimization", Parelec, 2006.
- [7] Riley, G. F., and M. H. Ammar, "Simulating large networks – how big is big enough?", Proceedings of First International Conference on Grand Challenges for Modelling and Simulation, January 2002.
- [8] Tim Dörnemann, Matthew Smith, Bernd Freisleben: Composition and execution of secure workflows în wsrf-grids, Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid(CCGRID), 122-129 (2001)
- [9] J. Basney, M. Humphrey, and V. Welch: The MyProxy Online Credential Repository, în Soft-ware, Practice and Experience, volume 35, 801-816 (2005).
- [10] Martin C. Brown: Build grid applications based on SOA, IBM (2007)
- [11] Jini™ Architectural Overview: www.sun.com/software/jini/whitepapers/architecture.pdf
- [12] Scott Oaks and Li Gong, Jxta în a Nutshell, O'Reilly & Associates Inc., 2002
- [13] Rajkumar Buyya, Economic-based Distributed Resource Management and Scheduling for Grid Computing, Ph. D. Thesis, Monash University, Melbourne, Australia, 2002.
- [14] T.L. Casavant, J.G. Kuhl. A Taxonomy of Scheduling în General-Purpose Distributed Computing Systems. în IEEE Transaction on Software Engineering, v.14 n.2, pp.141-154, 1988.
- [15] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 15 (3) pp 200-222, 2001. <http://globus.org/research/papers/anatomy.pdf>.
- [16] I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid - An Open Grid Services Architecture for Distributed Systems Integration (extended version of Grid Services Architecture for Distributed Systems Integration. IEEE Computer 35(6), pp 37-46). Open Grid Service Infrastructure WG, Global Grid Forum. 2002. <http://www.globus.org/research/papers/ogsa.pdf>.



UNIUNEA EUROPEANĂ



GUVERNUL ROMÂNIEI



Instrumente Structurale
2007 - 2013

- [17] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt. Open Grid services infrastructure (ogsi) version 1.0. 2003. http://globus.org/alliance/publications/papers/Final_OGSI_Specification_V1.0.pdf

