RAPORT INTERMEDIAR DE ACTIVITATE (RIA)

Denumirea Proiectului: "*GEEA – Centru de resurse GRID multi-corE de înaltă pErformAnţă pentru suportul cercetării, dezvoltării tehnologice şi inovării ştiinţifice pe plan European*"

Contract nr: 51/11.05.2009

Sursa de finanţare: POS CCE O2.2.3

Data finalizării: 10.05.2011

Cuprins

Activitatea 3.3. Stabilirea priorităţilor în raport cu nevoile specialiştilor şi utilizatorilor sistemului GRID naţional

Activitatea 3 a proiectului GEEA este realizarea de aplicaţii informatice necesare pentru administrarea şi funcţionarea centrului de resurse Grid al Universităţii Politehnica din Bucureşti (UPB) şi pentru optimizarea funcţionării Grid-ului la nivel naţional. În cadrul acestei activităţi, în prima fază au fost proiectate aplicaţiile ce vor fi dezvoltate prin explorarea proiectelor similare la nivel european şi internaţional pentru a asigura interoperabilitatea cu acestea. Utilizarea capacităţilor locale de a dezvolta aplicaţii este importantă pentru stabilirea priorităţilor în raport cu nevoile specialiştilor şi utilizatorilor sistemului GRID naţional. Activitatea va consta în fazele următoare în dezvoltarea aplicaţiilor identificate în raport cu priorităţile identificate şi capacităţile stabilite precum şi testarea aplicaţiilor şi realizarea documentaţiilor de utilizare. De asemenea stabilirea continuă a capacităţilor de a dezvolta aplicaţii va conduce la posibilitatea de creare de colaborări pe componenta de dezvoltare de aplicaţii. Prin stabilirea capacităţii de a dezvolta aplicaţii se asigură şi perfecţionarea echipei existente.

În vederea stabilirii priorităţilor de cercetare pentru proiectul GEEA, au fost analizate împreună cu principalii parteneri şi beneficiari ai proiectului direcţiile principale evoluţie a sistemelor de tip Grid şi a aplicaţiilor dedicate. Principalii beneficiari ai proiectului sunt: Consorţiul RoGrid, Facultatea de Automatică şi Calculatoare din cadrul Universităţii „Politehnica” din Bucureşti, Oficiul pentru Administrare şi Operare al Infrastructurii de Comunicaţii de Date RoEduNet.

Obiectivele principale ale proiectului GEEA sunt corelate cu necesităţile la nivel naţional cu privire la sistemele Grid. UPB are numeroase parteneriate în proiectele bazate pe sistemul GRID, cu instituţii care au participat iniţial la constituirea consorţiului RoGrid.

Stabilirea priorităţilor în raport cu nevoile specialiştilor şi utilizatorilor sistemului GRID naţional au fost realizate în colaborare cu principalii parteneri utilizatori de Grid din Romania.

Parteneri şi utilizatori ai sistemelor Grid

Institutul Naţional 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 tradiţie şi o prezenţă activă de peste 35 de ani în informatica românească, reprezintă cel mai important institut de cercetare-dezvoltare şi inovare în domeniul Tehnologiilor Informaţiei şi Comunicaţiilor din România. Prin asumarea priorităţilor ştiinţifice şi tehnologice ale domeniului,misiunea ICI o constituie consolidarea poziţiei

câştigate şi dezvoltarea suportului ştiinţific şi tehnologic în domeniul tehnologiilor informaţiei şi comunicaţiilor necesar realizării structurilor şi serviciilor specifice societăţii informaţionale bazate pe cunoaştere. Arii de competenţă ale ICI sunt orientate în domenii precum: reţele de comunicaţie şi tehnologii avansate pentru dezvoltarea de aplicaţii în medii distribuite; tehnologiile informaţiei şi comunicaţiilor în domenii de interes public; conţinut digital, creativitate şi dezvoltare personală; tehnologiile informaţiei şi comunicaţiilor, suport pentru dezvoltarea afacerilor şi a industriei; noi modele şi tehnologii în ingineria sistemelor şi a produselor software; sisteme bazate pe cunoştinţe, învăţare şi sisteme cognitive; sisteme avansate de calcul şi control. Portofoliul de produse create în Institut sau în colaborare cu parteneri interni sau externi constă din: aplicaţii de bioinformatică, modele avansate de decizie, modele de business, e-business, e-commerce, modele şi sisteme de diagnoză şi evaluarea riscului, modele, algoritmi şi tehnologii GRID, platformă GRID, platforme pentru promovarea afacerilor electronice, produse software pentru educaţie asistată de calculator.

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

Institutul Naţional de Fizica a Pământului (INFP). Principalele activităţi ale INFP sunt legate de monitorizarea seismică. Reţeaua Seismica Naţionala are în componenţă 73 de staţii seismice digitale în timp real (499 de canale la 100 eşantioane / canal) instalate intre 1995 pana în prezent. Staţiile seismice sunt echipate cu digitizoare Q330 şi K2, senzori de acceleraţie (EpiSensor, 2g) şi senzori de viteza (de banda larga şi scurta perioada). Cele mai multe dintre staţii sunt amplasate în estul şi sudul Carpaţilor fiind destinate în primul rând monitorizării seismice a zonei Vrancea. Transmisia de date dintre staţiile seismice digitale şi Centrul National de date de la Bucureşti se face prin diferite sisteme de comunicaţie (GPRS, VPN, satelit, radio, linii închiriate şi Internet), astfel încât back-up în cadrul reţelei s-a realizat prin distribuirea sistemelor de comunicaţie. Datele digitale de la staţiile seismice sunt analizate în timp real de un sistem automat de achiziţie şi prelucrare automata. Reţeaua Seismica Naţionala distribuie rezultatele sale (localizări şi fazele cutremurelor), prin Internet, către mai multe servicii seismologice din Europa.

O altă direcţie importantă se referă la sistemele complexe de monitorizare a Pământului. Sistemele complexe de monitorizare şi prelucrare în timp real şi cvasi-real sunt destinate urmăririi factorilor precursori ai evenimentelor seismice, fapt care conduce la alcătuirea unei bănci de date ce poate fi analizata în paralel cu activitatea seismica în vederea elaborării unei metode eficiente de prognoza pe termen scurt a activităţii regiunii Vrancea. Factorii precursori avuţi în vedere sunt: activitatea seismica în domeniul micro-cutremurelor, variaţia câmpului magnetoteluric corelată cu activitatea solară, variaţia concentraţiei gazelor alfa radioactive în lungul faliilor, ionizarea atmosferica în zona centrală, măsurători de emisie acustica în foraje, variaţia curenţilor telurici, variaţia nivelului pânzei freatice precum şi alte fenomene şi observaţii considerate a avea legătura cu activitatea seismică. Pentru fiecare dintre factorii enumeraţi mai sus INFP a realizat un sistem avansat şi inteligent de detecţie, de procesare şi de analiză a datelor experimentale utilizându-se o bază comuna de timp şi echipamente compatibile cu cele existente în Reţeaua Naţionala de supraveghere seismica.

Proiectele de cercetare în care institutul este implicat sunt: Cercetări fundamentale de seismologie istorică şi paleo-seismologie necesare pentru evaluarea seismicităţii pe termen lung şi a hazardului seismic, Macro-zonarea seismic a teritoriului României, fundamentată pe intensităţi macro-seismice reevaluate coroborate cu date geologice şi geofizice complexe, Reducerea riscului seismic prin evitarea rezonantei teren-structura şi prin izolarea bazei structurii. Aplicabilitate în zona metropolitana Bucureşti, Modelarea efectelor seismice locale induse de cutremurele crustale produse în zonele Tulcea, Galaţi - Tecuci, Câmpulung, Banat şi 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, asistenta, consultanta, experize privind metalurgia extractiva a metalelor neferoase şi rare, aliajelor, pulberilor, materialelor ceramice şi compozite, stocarea şi prelucrarea subproduselor şi deseurilor solide, valorificarea metalelor neferoase şi 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 în very aggressive environments, Hybrid catalysts with oxo(peroxo) metal species obtained by functionalization of mezostructured 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 Aerospaţiale "Elie Carafoli" (INCAS). INCAS este singura unitate din România care desfăşoară activitate de cercetare pe întregul ciclu, pornind de la cercetarea fundamentala de baza şi orientata, continuând cu cercetarea aplicativa şi terminând cu dezvoltarea tehnologica şi implementarea rezultatelor obţinute în producţie. Cercetarea fundamentala pe care o desfăşoară INCAS urmăreşte creşterea nivelului de cunoştinţe în domeniul ştiinţelor aerospaţiale şi aeronautice, referindu-se la: aerodinamica generala, dinamica zborului şi sistemelor, structuri aerospaţiale, aeroelasticitate, rezistenta materialelor utilizabile în aeronautica, sisteme de propulsie aerospaţiala. Cercetarea aplicativa - dezvoltarea tehnologica, care intra în profilul institutului, se refera la elaborarea de: tehnologii şi materiale aerospaţiale; echipamente electronice, mocano-hidraulice şi pneumatice; realizarea de modele experimentale din domeniul aeronautic şi aerospaţial; standuri şi instalaţii de testare; platforme şi staţii pilot; aparatura de laborator, scule şi dispozitive utilizabile în industria aeronautica. Proiectele acestui institut sunt: Dezvoltarea Mock-UP virtual şi realizare specimen proba statica pentru aeronava de transport aerian regional, Sistem sinergic pentru lansarea/recuperarea dinamica a obiectelor satelitare - SILROS, Structuri Morphing pentru aplicaţii în construcţii AeRospaŢiale – SMART, Materiale avansate pentru industria aerospaţiala: Compozite C-C cu matrice mezofazică aditivă cu nano-tuburi de carbon şi compozite laminate fibre - metal COMPAS.

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

Administraţia Naţionala de Meteorologie (ANM). Obiectul principal de activitate al Administraţiei Naţionale de Meteorologie îl constituie activitatea de meteorologie şi climatologie necesara dezvoltării social-economice a României şi integrarea acestei activităţi în sistemul de convenţii şi relaţii internaţionale, în care scop: efectuează observaţii şi măsurători asupra stării şi evoluţiei vremii; asigura şi controlează circulaţia internă de date şi informaţii meteorologice; elaborează metodologii de măsurare şi prelucrare a datelor şi realizează produse meteorologice; elaborează diagnoze, prognoze şi avertizări meteorologice; realizează studii de cercetări în domeniile meteorologiei; expertizează, validează şi stochează date şi informaţii specifice Fondului naţional de date meteorologice, în vederea tinerii la zi şi constituirii băncilor de date tematice la nivel naţional; elaborează studii şi scenarii de evoluţie a climei; asigura schimbul de date şi de informaţii meteorologice în flux rapid; realizează schimbului internaţional de date în domeniul meteorologiei; elaborează componentele de climatologie din studiile de impact şi de bilanţ de mediu; organizează şi coordonează sistemul naţional de formare şi pregătire profesionala în domeniul meteorologiei, climatologiei şi fizicii atmosferei. O componenta importanta a activităţii de meteorologie o constituie prelevarea şi transmiterea datelor de observaţii din reţeaua de staţii de suprafaţa, radar şi aerologice. Prin natura sa, meteorologia impune o strânsă colaborare internaţională, Romania fiind membru fondator al Organizaţiei Meteorologice Mondiale (OMM). Pe lângă schimbul extern de date, Administraţia Naţională de Meteorologie participa la acţiuni internaţionale din cadrul programelor OMM, fie legate de proiecte ştiinţifice finanţate de Comunitatea Europeana (CE) sau colaborări bilaterale şi multilaterale. Printre proiectele în care este implicată administraţia naţională de meteorologie amintim: cercetării climatologice, Fizicii Atmosferei şi Poluării Aerului, Meteorologie Dinamica, Climatologie şi Agrometeorologie, Metodică meteo, Modelare numerică, Prognoză de lungă durată, Teledetecţie.

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

Direcţii de cercetare identificate

Obiectivul 2 al proiectului GEEA constă în suportul aplicaţiilor inovative ale instituţiilor de cercetare naţionale pentru creşterea performanţei şi accesibilităţii sistemului naţional GRID şi a centrului de resurse GRID local. Pe de o parte un rol important îl au aplicaţiile pentru suportul asigurării interoperabilităţii, accesibilităţii, pentru monitorizarea şi controlul sistemelor GRID, pentru utilizarea şi partajarea optimă a resurselor. Aceste aplicaţii asigură suportul pentru un sistem distribuit de mari dimensiuni de încredere care oferă suportul aplicaţiilor complexe şi susţinerea calculului de înaltă performanţă. Pe de altă parte, suportul pentru aplicaţiile dezvoltate de centrele de cercetare locale descrise în secţiunea anterioară (Institutul de Fizica Chimie "Ilie Murgulescu", Institutul Naţional de Cercetări Aerospaţiale "Elie Carafoli", Administraţia Naţionala de Meteorologie *etc.*) oferă posibilitatea dezvoltării

acestora pe Grid, folosind un set de resurse de înaltă performanţă. Sunt aduse astfel contribuţii la obiectivul competiţiei de dezvoltare a sistemului Grid naţional prin soluţiile oferite pentru sporirea performanţelor.

Activitatea 3 susţine realizarea obiectivului 2 al proiectului prin dezvoltarea de aplicaţii relevante pentru optimizarea funcţionării sistemului Grid naţional prin aplicaţii de monitorizare a sistemului Grid creat, aplicaţii de simulare a sistemelor Grid în vederea optimizării aplicaţiilor, aplicaţii de planificare a aplicaţiilor în vederea optimizării timpului de acces la resurse şi a echilibrării încărcării resurselor, aplicaţii pentru controlul şi toleranţa la defecte a sistemului Grid, publicarea pe site-ul proiectului a unui set de ghiduri de utilizarea a aplicaţiilor create, ghiduri utile dezvoltatorilor de aplicaţii 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 cat ăi potențialii utilizatori accesează sistemul cu o frecvență ridicată, definește cerințe complet noi pentru strategiile de monitoriza 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 viabila de monitorizare în astfel de medii dinamice, în cadrul proiectului s-au utilizat informații provenite direct din sistemele de operare. O alta 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 şi autonomia site-urilor componente sunt aspecte de baza în mediile Grid. Datorita acestor aspecte, este dificil de experimentat şi testat algoritmii de planificare, deoarece prezenta unui număr mare de utilizatori face aproape imposibila 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 algoritmii de planificare. Instrumentul de simulare dezvoltat în cadrul acestui proiect oferă facilități diverse, cum sunt simularea execuției aplicațiilor, a transferurilor în rețea şi 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 sa descrie algoritmii, implementând o interfață specifica. Astfel au fost simulate şi testate atât strategii clasice de planificare, cat şi 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 dificila. Costul găsirii unei soluții optime pentru planificare este în cele mai multe cazuri prohibitiv, motiv pentru care sunt de preferat euristicile care ajung la soluții apropiate de cea optima. Un exemplu de eurisitca este cel propus în cadrul acestui proiect, bazat pe

algoritmi genetici. Testele efectuate au arătat ca algoritmii 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 facila. O alta abordare pe care o propunem este cea bazata pe modelul economic, având în vedere similitudinile dintre mediile Grid şi sistemul economiei de piața. Ca şi 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 orientata pe servicii a început sa se contureze recent şi a avut o evoluție rapida, fiind adoptat în tot 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 introdusa cu peste 20 de ani în urma prin programarea orientata pe obiecte. De asemenea, comunicarea intre aplicații/servicii, pe baza unui anumit protocol, a fost implementata prin mecanisme ca DCOM şi CORBA încă de când a început evoluția sistemelor distribuite.

Activitatea 3.1. Proiectarea aplicaţiilor ce vor fi dezvoltate prin explorarea proiectelor similare la nivel european şi internaţional, pentru a asigura interoperabilitatea

Activitatea 3.1 are ca scop identificare modelelor ştiinţifice şi proiectarea aplicaţiilor ce vor fi dezvoltate. S-au analizat principiile, metodele, modelele şi proiectele existente la nivel internaţional şi au fost identificate posibilităţile de utilizare. Astfel, vor fi descrise componentele de monitorizare, de simulate şi planificare în sistemele distribuite. La finalul secţiunii vom prezenta modelul SOA pentru realizarea aplicaţiilor.

Monitorizarea în sistemele distribuite

Soluţii 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 evoluţia în timp a sistemului. Informaţiile de monitorizare sunt importante în procesele de decizie pentru partea de planificare, de analiză a performanţelor sistemelor şi aplicaţiilor. Vom descrie pe scurt modelele arhitecturale folosite în prezent pentru monitorizarea sistemelor distribuite pe scara larga. Aceste solutii se circumscriu unui model generic structurat pe trei niveluri ale monitorizarii: nivelul inferior, al colectarii datelor de monitorizare reprezentat de ansamblul senzorilor care furnizeaza informatiile de interers despre sistemele observate, nivelul intermediar, al gestiunii datelor colectate şi nivelul superior, al prezentarii informatiilor de monitorizare utilizatorilor interesati.

Pentru a reliza interoperabilitatea intre diversele sisteme existente, Open Grid Forum a introdus un set de standarde şi modele arhitecturale pentru diferitele aspecte ale mediilor Grid. Intre acestea se gaseste GMA (Grid Monitoring Architecture), o arhitectura de baza pentru monitorizare în Grid, bazata pe interactiunea dintre trei componente: producatori, consumatori şi servicii de repertoriere. Sistemul de monitorizare R-GMA (Relational GMA) este o implementare a arhitecturii GMA care are doua proprietati specifice: nu este necesara cunoasterea serviciului de repertoriere (directory service) pentru a furniza sau obtine informatii (deoarece comunicatia cu serviciul este gestionata de producator şi consumator) şi intregul system functioneaza şi poate fi interogat ca o baza de date relationala. O alta solutie de monitorizare existenta care implementeaza GMA este GridICE, un proiect care a evoluat în cadrul EGEE. Acesta foloseste o ierarhie pe doua niveluri pentru colectarea datelor: colectare locala site-ului (cluster-ului) şi colectare la nivel de Grid. GridICE furnizeaza mai multe moduri de accesare a informatiilor de monitorizare: o interfata web cu reprezentari textuale şi grafice, o reprezentare XML care poate fi folosita de aplicatii dar şi un mecanism de publish / subscribe pentru notificari ale evenimentelor detectate.

Descrierea soluției de monitorizare propuse. Serviciul de Monitorizare a Grid-ului are scopul de a colecta informaţii privind starea Grid-ului, acesta fiind un mediu eterogen şi dinamic. în acest sens solutia propusa se bazeaza pe sistemul distribuit de monitorizare MonALISA împreuna cu ApMon, o aplicaţie de monitorizare care poate fi configurat pentru a trimite informaţii de stare în forma unor datagrame UDP către MonALISA. Sunt obţinute astfel informaţii despre noduri şi despre clustere, informaţii de reţea (trafic, conectivitate, topologie etc.) pentru WAN şi LAN, informaţii de monitorizare a performantelor aplicaţiilor, joburilor şi serviciilor. Informaţiile de monitorizare despre nodurile din cluster sau Organizatia Virtuala vor fi preluate de către Componenta de Planificare pentru a optimiza algoritmii folositi în functie de informatiile primite în timp real de la sistemul de monitorizare. Pentru colectarea informaţiilor, se va implementa o varianta a Clientului de MonALISA, care se conectează la serviciul de monitorizare prin servere proxy şi obţine date pentru algoritmii de planificare folositi. Astfel se introduc în calcule informaţii recente despre starea Grid-ului, pentru a obţine timpi de execuţie estimati cat mai aproape de cei reali.

Sistemul MonALISA este proiectat ca un ansamblu de subsisteme autonome, multi-thread, autodescriptive şi bazate pe agenti care sunt inregistrate ca servicii dinamice, şi sunt capabile sa colaboreze şi sa coopereze în realizarea unei strangeri masive de informatie şi sarcini de procesare. Acesti agenti pot analiza şi procesa informatia, intr-un mod distribuit, pentru a asigura optimizarea deciziilor în aplicatii la o scara larga. O arhitectura bazata pe agenti ofera abilitatea de investire a sistemului cu grade crescatoare de inteligenta, pentru a reduce complexitatea şi a face sisteme globale administrabile în timp real. Scalabilitatea sistemului deriva din folosirea motorului de executie multithreaded pentru gazduirea unei varietati de servicii dinamice sau agenti şi abilitatea fiecarui serviciu sa se inregistreze la randul sau şi apoi sa fie descoperit şi folosit de orice alte servicii, sau clienti care solicita asemenea informatie. Sistemul este proiectat sa integreze cu usurinta uneltele existente de monitorizare şi procedurile şi sa ofere aceasta informatie intr-un mod dinamic şi obisnuit oricaror alte servicii sau clienti. Scalabilitatea unui sistem deriva din uzul unui motor multi threaded pentru a gazdui o varietate de servicii dinamice, abilitatea fiecarui serviciu sa inregistreze la randul sau şi apoi sa fie descoperit şi folosit de orice alt serviciu sau clienti care solicita asemenea informatie. Mediul integreaza mai multe proceduri menite sa colecteze parametrii ce descriu nodurile computationale, aplicatiile şi performanta retelei. Agentii specializati mobili sunt folositi în cadrul MonALISA pentru a realiza sarcinile de optimizare globala optimization sau a ajuta şi a imbunatati operatia pentru sisteme distribuite mari prin realizarea unor sarcini de supraveghere pentru diferite aplicatii sau parametrii în timp real. MonALISA ruleaza în prezent pe aproximativ 360 de site-uri.

Folosirea ApMon în cadrul solutiei propuse permite mai departe particularizarea parametrilor monitorizati în functie de aplicatia tinta, extinzand astfel setul standard de parametrii (ex: load, memori libera disponibila, spatiu pe disc) cu parametrii specifici fiecarei aplicatii monitorizate. API-ul simplu pus la dispozitie permite integrarea facila cu orice aplicatie şi injectarea transparenta a parametrilor în sistemul MonALISA. Pentru vizualizarea statisticilor globale, de istorie sau timp real cu parametrii monitorizati se va folosi clientul de tip Repository. Acesta permite stocarea optimizata a datelor colectate de la toate serviciile

distribuite, accesarea şi vizualizarea lor, exportarea în alte aplicatii folosind interfete web, wap sau servicii web, agregarea datelor şi automatizarea deciziilor în functie de politicile specificate de utilizatori. Detaliem în sectiunile urmatoare cele 2 componente folosite pentru procesul de monitorizare Grid: ApMon şi Repository-ul MonALISA.

ApMon este un set de biblioteci flexibile care pot fi utilizate de orice aplicatie pentru a transmite informatii de monitorizare serviciilor MonALISA. Datele de monitorizare sunt trimise sub forma unor datagrame UDP către una sau mai multe masini pe care sunt active servicii MonALISA. Aplicatiile pot raporta periodic orice tip de informatii pe care utilizatorul doreste sa le colecteze sau sa le monitorizeze, existand posibilitatea ca pe baza lor sa fie activati agenti de decizie sau sa fie declansate alarme ce semnaleaza conditii de eroare. Bibliotecile ApMon sunt disponibile pentru 5 limbaje de programare: C, C++, Java, Perl şi Python; pot fi utilizate atat în programe complexe de procesare de date, cat şi din scripturi sau programe utilitare. Principalele avantaje ale ApMon sunt: flexibilitatea, configurarea dinamica, performanta ridicata a comunicatiei, informatia este stocata intr-un mod structurat în bazele de date MonALISA.

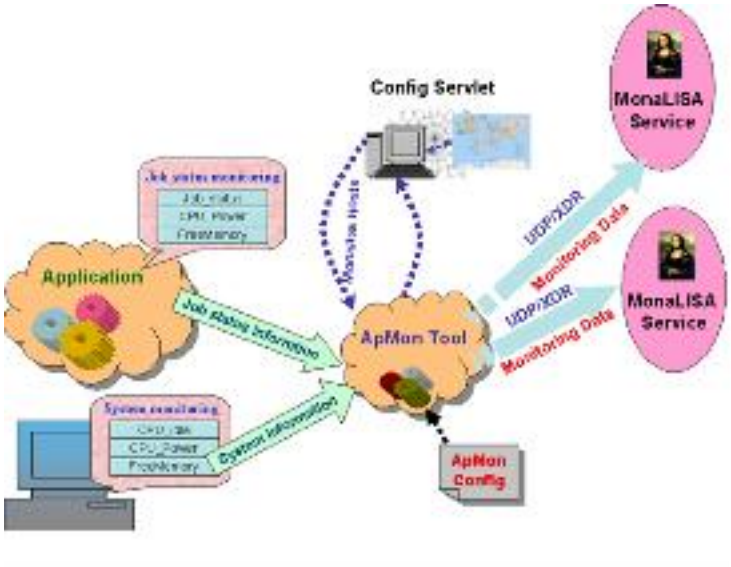

Figura 1. Scenariu de utilizare al aplicației ApMon

Procesul de instalare ApMon este simplu şi nu necesita alte componente software, ceea ce face ca acest instrument sa reprezinte o solutie rapida şi de cost scazut pentru a monitoriza atat job-uri, cat şi sisteme. Rutinele oferite de către biblioteca asigura codificarea datelor de utilizare în formatul XDR, precum şi construirea şi trimiterea de datagrame UDP. Dupa cum se observa în Figura 1 aplicatiile 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 performante ridicate a comunicatiei (intre o aplicatie instrumentata cu ApMon şi un serviciu MonALISA se pot transmite sute şi chiar mii de datagrame pe secunda). Pierderea ocazionala a catorva datagrame nu este fatala, deoarece scopul utilizarii este de a obtine informatii statistice; în plus, trimiterea se face în mod uzual la intervale de timp suficient de mici pentru ca pierderea unei datagrame sa nu fie critica.

Adresele serviciilor MonALISA către care se trimit date de monitorizare cu ApMon pot fi specificate fie în codul sursa al aplicatiei, fie în fisiere locale sau la distanta, accesibile prin HTTP. ApMon se poate auto-reconfigura în mod dinamic, reincarcand periodic fisierele şi paginile web de configurare. Pentru a evita problemele de securitate, serviciile MonALISA pot stabili o politica de a controla acceptarea mesajelor ApMon. Exista doua posibilitati de a stabili care datagrame ApMon sunt acceptate: fie prin stabilirea unei parole ce trebuie sa fie inclusa în datagrama, fie prin liste de adrese IP care sunt acceptate sau respinse. Aceste doua metode pot fi utlizate simultan.

Monitorizarea automata a parametrilor de sistem şi a job-urilor. Incepand cu versiunea 2.0, ApMon ofera posibilitatea de a transmite, dintr-un fir de executie separat, datagrame suplimentare ce contin informatii de monitorizare a sistemului, a aplicatiei care utilizeaza ApMon şi chiar şi a altor aplicatii. Datagramele pentru monitorizarea sistemului includ valori curente pentru parametri cum ar fi load-ul procesorului, numărul total de procese, cantitatea de memorie libera, spatiul 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 pana la momentul curent. Astfel de parametri sunt procentajul de timp în care procesorul a executat instructiuni în modul utilizator sau în modul sistem (raportat la timpul total), sau numărul mediu de KB transferati pe secunda prin fiecare interfata de retea. Exista şi parametri de sistem statici, cum ar fi numele sistemului, cantitatea totala de memorie, adresele IP sau frecventa procesorului. Datagramele pentru monitorizarea job-urilor contin valori pentru parametri cum ar fi cantitatea de memorie, spatiul pe disc şi timpul de procesor consumate. Se pot monitoriza job-uri multi-proces, în acest caz facandu-se o suma a cantitatilor de resurse consumate pentru toate subprocesele

Repository MonALISA. Mediul de monitorizare MonALISA ofera un mecanism simplu de a crea clienti capabili sa foloseasca mecanismul de descoperire din JINI şi sa gaseasca toate serviciile care ruleaza. Acesti clienti se pot inregistra la un set de parametrii sau agenti de filtrare pentru a primi anumite informatii de la toate serviciile. Aceasta ofera posibilitatea de a prezenta vizualizari globale ale ansamblului dinamic de servicii care ruleaza intr-un mediu distribuit, pentru a fi folosite de serviciile de nivel superior. Valorile primite sunt stocate local intr-o baza de date relationala (mySQL sau Postgres), asupra lor aplicandu-se proceduri dedicate de compresie pentru datele mai vechi.

Informatia astfel stocata e folosita pentru a crea un repository web, capabil sa ofere o viziune sintetica asupra modului de comportare al sistemelor distribuite la scara mare. Un mecanism bazat pe servleti e folosit pentru a prezenta grafice de timp real, de istorie, statistici, tabele de o maniera flexibila. Odata pornit, repository-ul se inregistreaza cu anumite predicate şi stocheaza valorile primite intr-o baza de date. Un predicat are urmatoarea forma: Ferma / Cluster / Nod / timp_start / timp_final / lista de functii. Acesti parametrii sunt ultrior afisati folosind mecanismul de servleti intr-o larga varietate de grafice, tabele statistice, bazate pe fisierele de configurare care descriu vizualizarile dorite şi parametrii proprii acestora, oferind astfel perspective globale sau particulare.

Repository-ul foloseste de asemenea o harta interactiva pentru vizualizarea pozitionarii fermelor dar şi pentru monitorizarea anumitor servicii, legaturilor sau traficului. Astfel este oferita functionalitatea clientului interactiv disponibil în MonALISA intr-o interfata adaptata web, specializata în interpretarea datelor. Figura 2 prezintă o perspectiva schematica asupra interactiunii dintre aceste servicii:

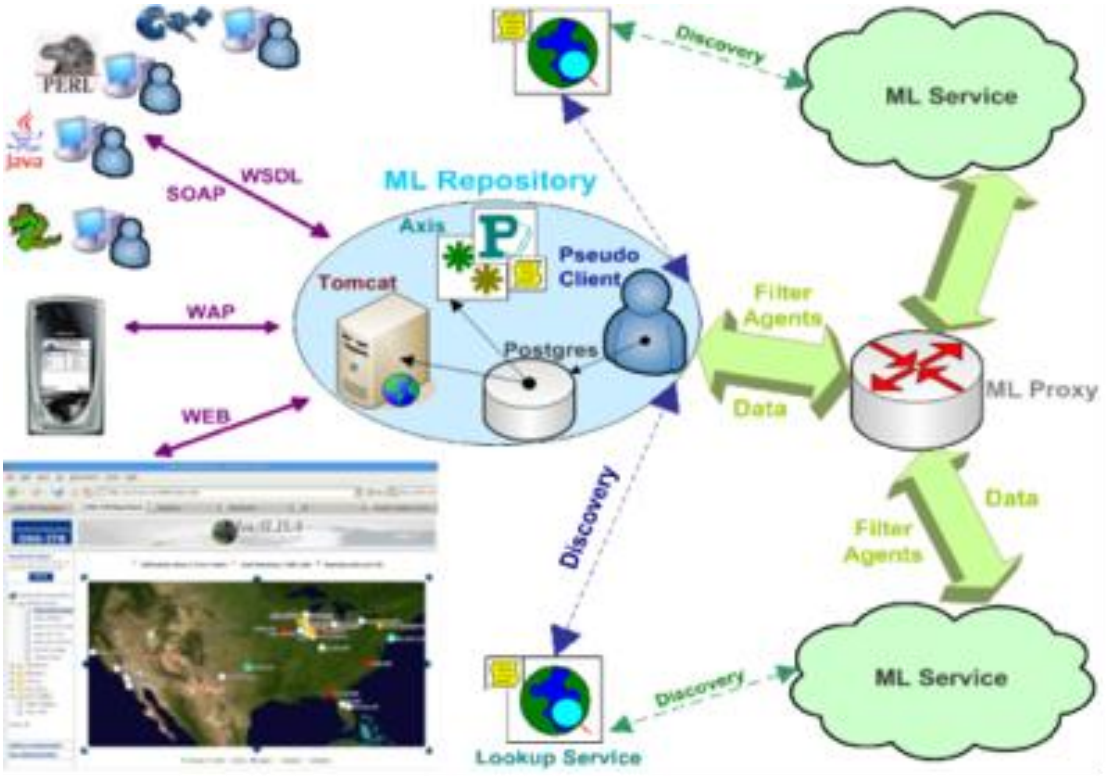

Figura 2. Arhitectura Repository-ului MonALISA

Bazat pe datele stocate în baza de date, şi pe fisierele de configurare se pot construi tabele statistice sau grafice de diverse tipuri: barchart, serii, spidere, pie, axe multiple, tabele, serii cu evenimente asociate etc. Aceste grafice ploteaza parametrii monitorizati: incarcarea siturilor, număr de CPU, număr de joburi, tipuri de joburi, joburi idle, joburi running, joburi pe VO, utilizarea nodurilor, parametrii sistem (memorie, spatiu disc, etc), trafic WAN, latime de banda disponibila, pathload, statusul siturilor monitorizate, etc. oferind o viziune globala asupra sistemului distribuit pentru a putea lua decizii de optimizare, sau o integra intr-un sistem de triggerare bazat pe anumite evenimente. Asupra datelor se pot adauga diverse niveluri de agregare, selectabile de către utilizatori din pagina de web: sumare, mediere, integrare, scari logaritmice pentru o mai buna intelegere a informatiei afisate. Utilizatorul dispune de asemenea şi de optiuni asupra modului de afisare: selectia tipului de grafic, a perioadei de plotare, selectia fermelor, a statisticilor generate pe baza graficului selectat, etc.

Fiind folosit pentru stocarea şi interpretarea datelor de la toate serviciile pe perioade mari, baza de date ajunge rapid la proportii mari iar query-urile adresate ei prin mecanismul de servleti pot deveni astfel consumatoare de timp. Pentru a rezolva aceasta problema 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 dimensiunue adaptabila (fereastra de timp acoperita difera în functie de memoria disponibila) de unde sunt rapid servite servletilor, inainte 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 resultui 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 algorimi de detectie de pattern-uri, stabilire de corelatii intre evenimentele monitorizate şi folosirea acestor informatii pentru automatizarea procesului de gestiune a resurserlor şi de administrarea a Organizatiilor Virtuale.

Simularea sistemelor distribuite

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

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

Prin comparaţie cu alte instrumente de simulare ce se adresează sistemelor distribuite, soluţia prezentată în cadrul acestei lucrări oferă o serie de avantaje. Una dintre problemele cu care se confruntă astfel de aplicaţii ţine de scepticismul utilizatorilor în legătură cu validitatea rezultatelor obţinute. MONARC a fost testat cu succes în diverse experimente ce au urmărit atât verificarea validităţii modelului folosit şi a soluţiilor de implementare, cât şi verificarea performanţelor 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 execuţiei proceselor) şi 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 reţea până la sisteme distribuite de mari dimensiuni cuprinzând o varietate mare de resurse sau aplicaţii 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 şi dinamism în evaluarea performanţelor unei game largi de arhitecturi de procesare. Simulatorul conţine mecanismele necesare modelării traficului concurent şi evaluării de diverse strategii de replicare a datelor sau de planificare a execuţiei sarcinilor de lucru. Caracteristicile aplicaţiei precum arhitectura robustă, implementarea modulară şi extensibilă, setul bogat de componente disponibile, fac din MONARC un instrument de modular deosebit de popular în rândul utilizatorilor din diverse domenii ştiinţifice. Folosirea tehnologiilor de monitorizare în procesele de validare şi proiectare au consolidat şi mai mult succesul de care se bucură acest proiect.

Componentele şi arhitectura simulatorului. Arhitectura aplicaţiei de simulare este special proiectată pentru modelarea unor sisteme distribuite de mari dimensiuni, cuprinzând interacţiuni complexe şi schimbul şi procesarea unor cantităţi impresionante de date. Cea mai potrivită soluţie 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 şi facilă a componentelor logice peste programul de simulare şi furnizarea mecanismelor de interacţiune necesare modelării unor sisteme distribuite de mare întindere (vezi Figura 3).

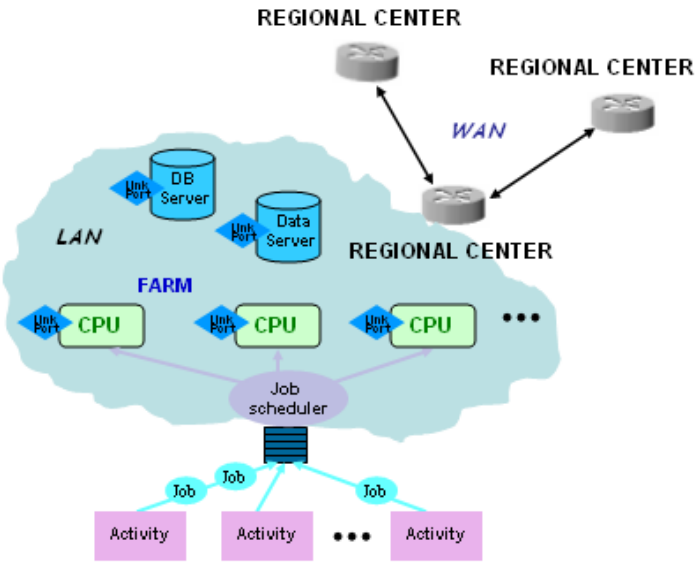

Figura 3. Componentele modelului de simulare

Simulatorul MONARC foloseşte o abordare orientată spre procese pentru implementarea simulării bazate pe evenimente discrete, aspect ce facilitează simularea aplicaţiilor concurente şi a modelelor stocastice. Obiectele ce folosesc fire de execuţie, numite şi „Obiecte Active” (ce folosesc fire de execuţie pentru implementarea secvenţei de rulare şi 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 şi distribuite orientate spre procese. Java include suport nativ pentru proiectarea multi-thread a procesării concurente, aspect ce la nivelul simulării a fost folosit în combinaţie cu implementarea unui mecanism de planificare dedicat. Motorul de simulare beneficiază astfel de proprietăţi ce îi permit instalarea uşoară şi rularea unor scenarii complexe chiar şi pe sisteme multi-procesor. O contribuţie importantă la dezvoltarea proiectului a fost adusă de numeroşii paşi de evaluare şi de optimizare executaţi ce au dus la creşterea performanţei deosebite a motorului de simulare.

Motorul de simulare. Nivelul de jos al arhitecturii aplicaţiei de simulare este reprezentat de motorul de simulare. Acesta este responsabil cu gestionarea firelor de execuţie, a evenimentelor de simulare, a cozilor de evenimente şi cu gestionarea mecanismului prin care task-urile interacţionează şi 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 foloseşte suportul nativ, superior în termeni de performanţă, al tehnologiei Java pentru procesarea concurentă. Tehnologia Java furnizează câteva avantaje: suport nativ pentru multithreading, paradigma programării orientate pe obiecte şi portabilitate. De altfel portabilitatea permite folosirea aplicaţiei de simulare pe o mare varietate de platforme (sisteme mono sau multi-procesor rulând Linux, Solaris sau Windows).

Task-urile de simulare pot gestiona execuţia activităţilor de simulare sau pot reprezenta componente ale sistemelor simulate. Un task de simulare reprezintă o entitate ale cărei proprietăţi se modifică dinamic în timp, comportamentul acestuia fiind influenţat direct de evenimentele de simulare. În cadrul simulatorului task-urile interacţionează prin schimburi de evenimente. Execuţia acestora este gestionată de către un planificator centralizat dedicat. Acest planificatorul are rolul de a gestiona atât execuţia task-urilor, cât şi a evenimentelor de simulare produse.

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

componente ce modelează comportamentul diverselor resurse de procesare, de date, de reţea şi pe cele ale diverselor activităţi corespunzătoare aplicaţiilor distribuite.
Arhitectura aplicaţiei de simulare este bazată pe modelul Centrului Regional. Centrul regional reprezintă un obiect complex compozit ce conţine servere de date şi noduri de procesare, toate conectate prin intermediul unei reţele locale. Un centru regional poate fi conectat cu alte centre prin intermediul ruterelor şi al reţelelor de mare întindere. De asemenea un centru regional poate instanţia dinamic un număr de obiecte ce modelează utilizatori şi activităţile 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. Aceştia pot genera sarcini de lucru de procesare de date pe baza unor scenarii. Fiecare centru regional poate avea una sau mai multe astfel de activităţi ce submit sarcini de lucru. Entitatea farm gestionează nodurile de procesare ale centrului regional. Rolul acesteia este de a aştepta submiterea de sarcini de lucru generate de activităţi şi de a apela planificatorul adecvat pentru execuţia acestora. Obiectul CPUUnit descrie comportamentul unei staţii de lucru aparţinând sistemului modelat. Acesta este caracterizat printr-o cantitate de putere de procesare şi 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 funcţie de un mecanism de priorităţi. Utilizatorul poate astfel specifica sarnici având priorităţi mai ridicate decât altele pentru execuţie.

O extensie a unităţii de procesare este reprezentată de CPUCluster. Acest obiect modelează un cluster de unităţi de procesare. În afară de puterea de procesare şi cantitatea de memorie disponibilă, el este caracterizat şi printr-o cantitate de spaţiu de stocare. Această clasă include mecanismele modelării inclusiv a defectării diverselor unităţi 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 folosita pentru a “gestiona eficient si intr-o maniera eficace accesul la folosirea resurselor Grid de către utilizatori” [13]. Conform [14] planificarea în sistemele Grid este compusa din trei componente principale: Consumatorii, Resursele şi Politica (vezi Figura 4).

Cele doua comunitati majore implicate în Grid, consumatorii de resurse care submit diverse aplicatii, respectiv furnizorii de resurse care isi partajeaza resursele, au de obicei motivatii diferite cand intra în Grid. Aceste motivatii sunt reunite în functiile obiectiv. în prezent, majoritatea functiilor obiectiv în Grid sunt mostenite din sistemele distribuite si paralele traditionale. Utilizatorii Grid sunt în principal interesati de performanta propriilor aplicatii, 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.

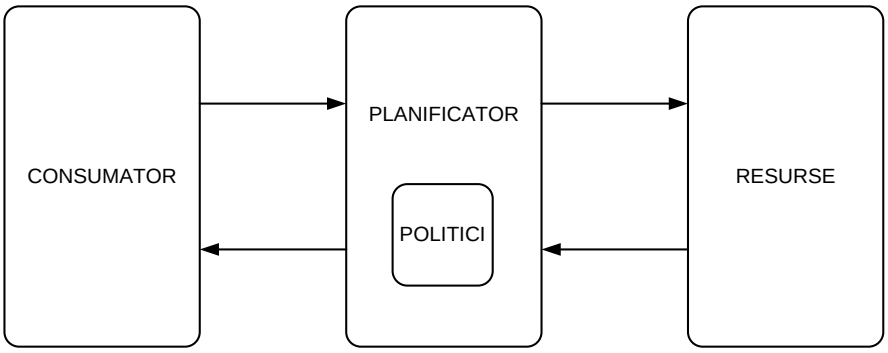

Figura 4. Modelul general de planificare

Obiective de planificare centrate pe aplicaţii. Acesti algoritmi incearca sa optimizeze performanta fiecarei aplicatii. Majoritatea obiectivelor aplicatiilor Grid curente sunt legate de timp, spre exemplu makespan, timpul trecut de la inceputul primului task dintr-un job pana la sfarsitul ultimului task din job. Pe masura modelele economice sunt introduse în Grid, costul economic al unei aplicatii devine o preocupare pentru utilizatori, respectiv costul platit de o aplicatie pentru a folosi o anume resursa. Pe langa aceste functii simple multe aplicatii folosesc functii compuse, spre exemplu unele doresc atat timp redus de executie si costuri economice mici. Dificultatea principala în adoptarea unor astfel de functii sta în normalizarea celor doua metrici diferite: timpul si banii. Odata cu dezvoltarea infrastructurii Grid s-a observat o tendinta orientata spre servicii, astfel incat calitatea serviciului (QoS) capata o importanta marita. Sensul QoS este dependent de aplicatii particulare, de la capacitati hardware la software. De obicei, QoS este o constrangere impusa proesului de planificare în locului functiei 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 numar de joburi intr-o perioada anumita de timp; utilizare – procentajul de timp în care o resursa este ocupata. O utilizare scazuta inseamna ca resursa este idle si astfel irosita. Pentru o resursa multiprocesor, utilizari diferite intre procesoare descrie de asemenea echilibrul incarcarii sistemului si o scadere a throughput-ului. Condor [Condor07] este un sistem care adopta throughput-ul ca politica de planificare. O alta politica viabila de administrare a resurselor o constituie profitul economic – beneficiul economic adus de atragerea aplicatiilor utilizatorilor Grid pentru a rula pe propriile resurse.

Functiile obiectiv mentionate anterior dau rezultate bune facand o presupunere, anume ca resursele sunt dedicate deci furnizeaza performanta constanta unei aplicatii. Insa aceasta presupunere nu tine intr-un mediu Grid asa cum am aratat în sectiunea precedenta. Spre exemplu, sa presupunem ca a fost gasita o planificare optima cu makespan OPT, daca

resursele implicate sunt stabile. Daca resursele Grid sunt brusc incetinite (interupte de joburi ale utilizatorilor locali, legatura la retea, etc.) si aceasta situatie persista pentru o perioada mai mare de timp, atunci makespan-ul planificarii reale este departe de OPT si nu poate fi limitat de algoritmi de planificare care nu pot prevedea schimbarile în performanta. Deci, daca functia obiectiv a unei planificari este makespan-ul si nu exista o limita fie pentru performantele resurselor fie pentru perioda de timp a executiei, cu alte cuvinte, daca nu putem prevedea fluctuatile de performanta, nu va exista în general un algoritm de aproximare makespan care sa fie aplicabil în Grid.

Pentru a elimina acest neajuns, a fost propus un nou criteriu de planificare: consum total de cicluri procesor (CTCP) – numarul total de instructiuni pe car Gridul le poate executa de la timpul de start al planificarii pana la terminare. CTCP reprezinta puterea de calcul totala consumata de o aplicatie. Cu acest nou criteriu, lungimea unui task este numarul de instructiuni din task; viteza unui procesor este numarul de instructiuni executate în unitatea de timp; iar procesoarele dintr-un grid sunt eterogene deci au viteze diferite. în plus, viteza fiecarui procesor variaza în timp datorita cursei intre procese în cadrul unui sistem deschis. Avantajul CTCP este ca e putin afectat de variatia performantelor resurselor dar totusi legat de makespan.

Structura sistemului de planificare dezvoltat în UPB este prezentată în Figura 5. Ţinând cont de sistemul distribuit real în care va funcţiona planificatorul şi datorită interacţiunii acestuia cu diferite alte aplicaţii (un exemplu ar fi folosirea datelor de monitorizare) arhitectura construita este formata din mai multe servicii. Componentele arhitecturii au ca rol descoperirea resurselor disponibile, rularea algoritmului de planificare, execuţia şi monitorizarea evoluţiei task-urilor în sistemul distribuit.

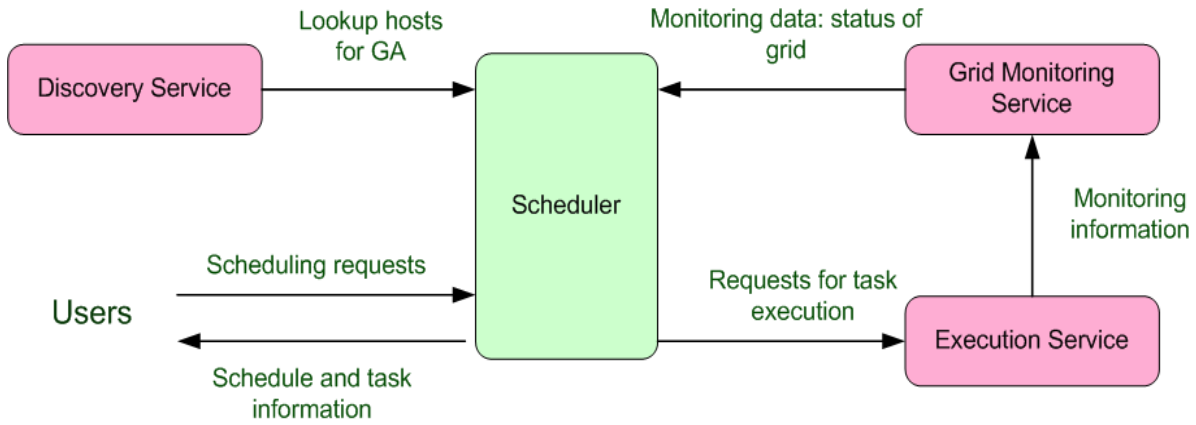

Figura 5. Structura sistemului de planificare DIOGENES

Componentele funcţionale ale sistemul, componente care interacţionează cu planificatorul sunt: Serviciul de Cereri; Serviciul de Monitorizare a Grid-ului; Serviciul de Execuţie; 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 construcţia modelului, sarcinile trebuie să respecte restricţii referitoare la timpii de execuţie, la cerinţele de viteza de procesare şi de memorie pentru sistemul pe care o aceasta va rula, ţinându-se totodată cont şi de ordonarea în timp a sarcinilor.

În cadrul planificatorului DIOGENES un scop important este îmbunătăţirea modului de descoperire a agenţilor. Deoarece este un planificator descentralizate acesta poate avea un rol foarte important pentru performanţa şi viteza cu care noii agenţi sunt integraţi şi pot primi cereri de planificare.

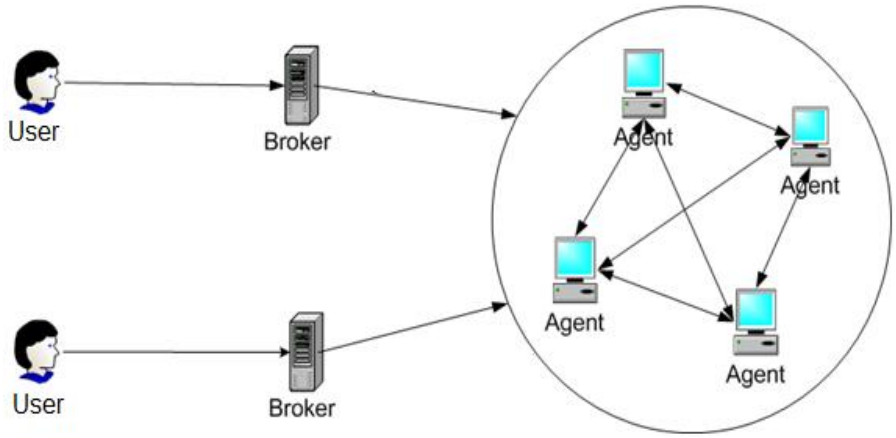

Figura 6. Modelul de comunicaţie în DIOGENES

Modelul actual al planificatorului, prezentat în Figura 6, se bazează pe tehnologia Jini [11]. Se foloseste astfel un serviciu central numit lookup unde se înregistrează toţi agenţii. Acest serviciu asigură şi comunicaţia între agenţi funcţionalitatea întregului sistem. Noii agenţi care intră în sistem trebuie să se înregistreze printr-un protocol de descoperire la serviciul de lookup. După acest pas agentul primeşte informaţii despre interfeţele implementate şi despre moduri de comunicare cu alţi agenţi din sistem. Acesta este o metodă nesigură deoarece serviciul de lookup este centralizat şi toţi agenţii şi-ar pierde funcţionalitatea în cazul în care acesta se defectează. În plus, un serviciu centralizat mai prezintă şi probleme de scalabilitate în cazul în care numărul de agenţi creşte foarte mult.

O soluţie 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 reţele punct-la-punct de mari dimensiuni. JXTA oferă de asemenea un limbaj abstract pentru modelarea comunicaţiei între agenţi. Se folosesc de asemenea mesaje în format XML pentru a putea fi uşor de împachetat, trimis şi pentru o compatibilitate mai bună cu diferite medii de transport. Ca un avantaj faţă de Jini, care foloseşte RMI şi serializarea obiectelor

Java pentru trimiterea de mesaje, formatul XML prezent în JXTA oferă o modularitate şi o independenţă faţă de limbajul de programare folosit.

Marele avantaj al acestei tehnologii este faptul că nu foloseşte un server centralizat pentru managementul agenţilor care rulează la un moment dat, cum este cazul sistemului Jini. Se permite astfel folosirea oricărui agent existent deja pentru a înregistra noii veniţi în sistem şi pentru a răspândi informaţii corecte despre toate serviciile disponibile. Se elimină astfel problemele de scalabilitate întâlnite daca ar fi fost nevoie interogarea unui server de fiecare dată când un agent dorea să cunoască starea altui agent din sistem. Se foloseşte 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.

Toţi agenţii rulează astfel independent, având posibilitatea să trimită mesaje de la unul la altul fără restricţii din partea infrastructurii şi la pornire şi înregistrare va cunoaşte în timp real starea oricărui alt agent din sistem. Comunicaţia se realizează prin canale de comunicaţie create special care se comportă ca linii seriale, indiferent de mediul de comunicaţie de la nivelele inferioare.

În plus, aceste avantaje uşurează foarte mult modul de programare a aplicaţiilor distribuite. JXTA oferă o abstractizare a arhitecturii sistemului distribuit în cadrul căreia rulează aplicaţia şi simplifică paşii necesari pentru coordonarea unui sistem de agenţi, multe din mecanismele necesare făcându-se automat, fără intervenţia dezvoltatorului.

Servicii Grid (implementarea SOA pentru Grid)

Arhitectura orientată pe servicii(SOA) se bazează pe conceptul de cuplare slabă. SOA defineşte un mod de dezvoltare a aplicaţiilor care se bazează pe împărţirea în servicii individuale, în loc să fie focalizate pe aplicaţii formate din mai multe funcţii [10].

Arhitectura orientata pe servicii în sistemele Grid. Avantajele arhitecturii orientate pe servicii au determinat adoptarea acesteia si în sistemele Grid, intr-un moment în care acest concept era încă nou. Orientarea pe servicii a fost o evolutie fata de primul model arhitectural al Grid-ului, propus de Ian Foster, Carl Kesselman si Steven Tuecke în [15]. Acest model implica existenta mai multor niveluri, aranjate în mod asemanator unei clepsidre: partea ingusta a clepsidrei corespunde unui set mic de protocoale si abstractii de baza; deasupra acesteia pot fi mapate o multime larga de comportamente, iar dedesubt pot exista tehnologii de nivel jos foarte diverse.

De la acest model bazat pe protocoale s-a fact trecere la unul orientat pe servicii. Astfel, [16] descrie Grid-ul ca fiind “un set extensibil de servicii care raspund la mesaje specifice protocoalelor”. Serviciile Grid pot fi agregate pentru a indeplini cerintele organizatiilor virtuale. Acest model nou, denumit “Arhitectura Deschisa 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 interfetei 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 inalt [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 intermedil retelei. Perspectiva orientata pe servicii simplifica virtualizarea (incapsularea diverselor implementari în spatele unei interfete comune) si raspunde la nevoia de a avea mecanisme standard de definire a interfetelor.

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. Definitiile 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 specificaţiile SOA. Grid-ul, fiind un sistem distribuit este perfect pentru natura distribuită a SOA, permiţând integrare uşoară, flexibilitate şi acces sigur. Pot apărea totuşi probleme de latenţă şi concurenţă. În cadrul SOA tehnologiile de obicei includ SOAP, XML şi standardele serviciilor web. XML formează partea de bază a tuturor standardelor, de la protocolul SOAP care este folosit pentru schimbul de informaţii 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 clienţii potenţiali.

Mai există şi un număr de probleme de securitate legate de folosirea arhitecturii orientate pe servicii în medii Grid. Exista 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 confidenţialitatea, sau WS-Trust, care oferă şi alte primitive foarte importante şi extensii pentru schimbul de tokenuri de securitate şi permite emiterea şi diseminarea de credenţiale 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 şi reprezintă una din soluţiile posibile pentru problema securităţii 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 proprietăţi 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. Proprietăţi de securitate pentru invocarea unui serviciu web

În mod normal când un client invocă un serviciu web sau trimite spre execuţie un flux acesta nu se află pe aceeaşi maşină 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 şi 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 maşina 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 soluţie foarte simplă ar fi stocarea credenţialelor utilizatorilor pe aceeaşi maşină pe care rulează portalul şi să se creeze certificatele Proxy atunci când este nevoie. Aceasta este o metoda nesigura

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 soluţie mai sigură ar fi adăugarea de informaţii despre credenţialele utilizatorilor în headerele cererilor clienţilor de tip SOAP. Această metodă este mai sigură deoarece doar clientul este deţinătorul cheii sale publice dar implică modificarea modului de invocare a serviciilor pentru a putea procesa şi aceste informaţii adiţionale.
În ambele cazuri se foloseşte serviciul MyProxy [9] de management al credenţialelor. Acesta este o aplicaţie folosită pentru managementul certificatelor de tip X.509 care permite generarea de certificate Proxy pe baza credenţialelor stocate ale utilizatorilor printr-o conexiune sigură de tip TLS. Utilizatorul va avea totuşi nevoie să creeze direct acest certificat Proxy care are o durată de viaţă scurtă. Deoarece timpul de rulare a unui serviciu invocat este necunoscut se vor crea şi mecanisme de reînnoire a certificatelor dacă acesta depăşeşte durata de viaţă a certificatului Proxy.

Soluţia prezentată oferă un grad înalt de securitate deoarece foloseşte 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 soluţie 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.

[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