Elektrotehniški vestnik 76(4): 193-198, 2009
Electrotechnical Review: Ljubljana, Slovenija

Razvoj storitve posredovanja ogrodnih virov po specifikacijah WSRF

Jernej Trnkoczy[1]

[1] *Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, SI-1000 Ljubljana, Slovenija*
E-pošta: jernej.trnkoczy@ldos.fe.uni-lj.si

Povzetek. Ena ključnih lastnosti ogrodnih sistemov je, da temeljijo na standardih, ki omogočajo globalno medobratovalnost storitvenih infrastruktur. Tako je večina današnjih ogrodij zasnovana na specifikacijah Web Services Resource Framework (WSRF), ki so trenutno najpomembnejši standard na tem področju. Omenjene specifikacije skupaj s še nekaterimi specifikacijami iz sklada spletnih storitev določajo standarden način za ustvarjanje, upravljanje, naslavljanje in obveščanje o primerkih stanja, ki pripadajo sicer brezstanjskim ogrodnim storitvam. V okviru projekta DataMiningGrid smo po specifikacijah OGSA WSRF Basic Profile 1.0 razvili storitev posredovanja virov, ki omogoča transparentno deljenje različnih vrst virov, ki pripadajo različnim administracijskim domenam in katerih zmogljivosti se s časom dinamično spreminjajo. Skladnost razvite storitve s standardi namreč zagotavlja medobratovalnost z drugimi storitvami, ki sodelujejo v procesu posredovanja virov. Implementacija je potekala na podlagi prilagojenega sistemskega programja Globus Toolkit četrte generacije (GT4), ki je de-facto standardno okolje za razvoj ogrodnih aplikacij.

Ključne besede: ogrodje, posredovanje virov, WSRF, posel, odprta ogrodna storitvena arhitektura

Developing a WSRF-compliant grid resource-broker service

Extended abstract.

The sharing of various kinds of distributed resources in heterogeneous and dynamically changing environments represents the core functionality of any grid system. The resources are shared among users with various levels of inter-trust-relations, and belonging to different administrative domains with different security policies. The sharing is achieved through software modules called resource brokers that hide the complexity of grids by automatic submission of the user's jobs to most suitable grid resources. The functionalities of resource brokers include transformation of user requirements into a set of grid jobs, examination of grid resources and their capabilities, matching of job requests and their requirements against the resources, scheduling of jobs on the appropriate resources, initiation and monitoring the execution of jobs and collection of results when they are finished.

In this paper, the development of a resource-broker service implemented under the DataMiningGrid [1] project will be presented. The project represents a large-scale effort aimed at developing a generic system facilitating the development and deployment of grid-enabled data mining applications. The project developed a grid middleware that supports execution of data mining tasks while utilizing data and computational resources of the grid. As a part of this middleware, a resource-broker service that takes into account special requirements of the data mining applications was designed and implemented.

One of the key requirements for the developed service was compliance with the existing grid standards. Standardization of the implementation technologies as well as the core grid functionalities are important for allowing interoperability among different grid components. Today, the Open Grid Services Architecture (OGSA) [2] specifications, which were published by the Open Grid Forum (OGF) [3], represent the most mature standards in the area of grid computing. OGSA aims to standardize practically all the services one commonly finds in a grid system by specifying a set of standard interfaces for these services. Furthermore OGSA defines extensions of the common web services technologies and in this way acts as a standard middleware between grid applications and low-level web services technologies [Figure 1].

Extensions of the basic web services specifications are needed since they do not provide a standard way of modeling and management of the service state. Guided by the principle of interoperation through conformance specification, OGSA therefore proposes a set of specifications known as OGSA WSRF Basic Profile 1.0 [4] that consists of a set of de facto, institutional or evolving web services specifications, along with clarifications, refinements, interpretations and amplifications of those specifications that promote interoperability among implementations of the basic profile. The profile includes specifications of interfaces and behavior defined in WSRF, WS-Addressing and WS-Notification families of specifications.

Globus Toolkit Version 4 (GT4) [5] represents a de-facto standard for developing grid applications. It includes quite a few high-level services that can be used to build grid applications, along with the service development and service execution environment. GT4 provides implementations of WSRF specifications for the programming languages Java, C and Phyton. The GT4 toolkit and Java WSRF implementation (Java WS Core) were used in implementation of the WSRF-compliant resource-broker service presented in this paper.

Keywords: grid, resource-broker, WSRF, job, Open Grid Services Architecture

Prejet 15. december, 2008
Odobren 17. april, 2009

1 Uvod

Skupna lastnost vseh ogrodnih sistemov je deljenje virov v porazdeljenih, raznolikih in dinamično spreminjajočih se okoljih. Viri so najpogosteje v ogrodju porazdeljene procesorske zmogljivosti in porazdeljen pomnilniški prostor za shranjevanje podatkov, nemalokrat pa tudi različni merilni instrumenti.

V želji po čim boljši izkoriščenosti virov je potrebna programska komponenta, ki na podlagi zahtev prihajajočih uporabniških poslov in trenutnega stanja ogrodnega sistema posle razvrsti na porazdeljene in raznorodne ogrodne vire. Razvrščanju pravimo posredovanje virov, komponentam, ki to funkcijo opravljajo pa posredniki virov. Pri tem je pomembno, da programske komponente posredovanja virov sledijo standardom, v nasprotnem primeru naj sistem ne bi bil pravi ogrodni sistem [6].

Implementacije različnih posrednikov virov so zelo raznolike, tako z arhitekturnega in funkcionalnega kot tudi s tehnološkega vidika. Med bolje poznane posrednike virov prištevamo programske komponente GridBus [7], GridWay [8], Nimrod-G [9] in GRMS [10]. V okviru projekta DataMiningGrid [1] smo razvili lastno komponento za posredovanje virov, ki je namenjena razvrščanju različnih, medsebojno neodvisnih opravil s področij ekološkega modeliranja, bioinformatike in digitalnih knjižnic. Lasten razvoj je bil potreben, saj nobena od že obstoječih rešitev ni sledila trenutno obstoječim ogrodnim standardom, poleg tega pa omenjene rešitve niso izpolnjevale vseh v projektu zastavljenih zahtev. Tako GRMS na primer ne podpira podajanja uporabniških zahtev v obliki nabora mogočih vhodnih parametrov, Nimrod-G in GridBus pa ne omogočata avtomatske izdelave seznama trenutno prostih virov v ogrodju.

V tem članku je predstavljen potek razvoja posrednika virov, pri tem pa je posebna pozornost namenjena razlagi uporabljenih standardov. V drugem poglavju je podan pregled uporabljenih standardov in tehnologij. V tretjem poglavju so predstavljene predvidene funkcionalne in arhitekturne zahteve do razvitega posrednika virov. V četrtem poglavju je predstavljena implementirana storitev. Ta je tudi umeščena v celotno arhitekturo sistema. Opis ocenjevanja razvite storitve je podan v petem poglavju. Sledi sklepno poglavje, v katerem so med drugim predlagane tudi mogoče izboljšave razvitega posrednika virov.

2 Izbrani standardi in tehnologije

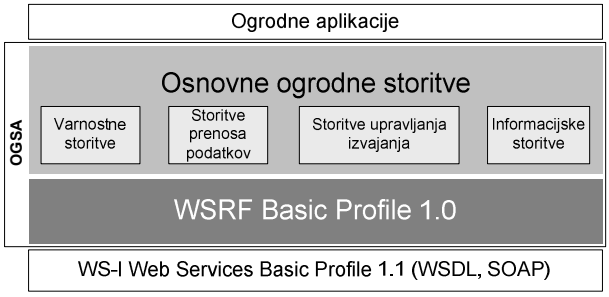
Zaradi velike kompleksnosti ogrodnih sistemov se pri njihovi implementaciji teži k ponovni uporabi že obstoječih komponent, ta pa je mogoča le v primeru njihove standardizacije. Prav sledenje standardom naj bi bilo ena od ključnih lastnosti, ki loči današnje tehnologije ogrodja od že dolgo obstoječih porazdeljenih sistemov [6].

Današnji ogrodni sistemi sledijo konceptu storitveno zasnovane arhitekture, v kateri so posamezne ogrodne komponente implementirane kot storitve. Trenutno edini poizkus standardizacije ogrodne storitveno zasnovane arhitekture so specifikacije odprte ogrodne storitvene arhitekture (Open Grid Services Architecture – OGSA) [2]. Ena najpomembnejših zahtev omenjene arhitekture je potreba po standardnih mehanizmih za ohranjanje in manipulacijo stanja storitev. Standardizacijo teh mehanizmov pomenijo specifikacije iz družine OGSA WSRF Basic Profile 1.0. V nadaljevanju bodo podrobneje predstavljeni omenjeni standardi, pa tudi med implementacijo uporabljeno razvojno okolje Globus Toolkit četrte generacije (GT4) [5].

2.1 OGSA in WSRF

Povezovanje med ogrodnimi storitvami je mogoče le, če je standardizirana tako nizkonivojska tehnologija, na kateri so storitve zasnovane, kot tudi funkcionalnost tistih ogrodnih storitev, ki jih za delovanje potrebuje večina ogrodnih aplikacij. Arhitekturo OGSA, delo organizacije Open Grid Forum (OGF) [3] torej predstavljajo specifikacije, ki se delijo na dva glavna dela (slika 1), in sicer:

1. infrastrukturni del, to je družina specifikacij WSRF Basic Profile 1.0 [4],
2. specifikacije funkcionalnosti osnovnih ogrodnih storitev.



Slika 1: Arhitektura OGSA
Figure 1: OGSA architecture.

Arhitektura OGSA kot temelj za implementacijo storitev predpisuje uporabo navadnih spletnih storitev (profili WS-I BP 1.1 in WS-I BP 1.0.). Vendar navadne spletne storitve nekaterih zahtev, ki nastajajo v ogrodnih sistemih, ne izpolnjujejo, zato je bila potrebna razširitev omenjenih standardov. To je družina specifikacij OGSA WSRF Basic Profile 1.0. Omenjeni profil združuje specifikacije WSRF, WS-Notification in WS-Addressing, ki definirajo standardne mehanizme za ustvarjanje, naslavljanje, spreminjanje vrednosti, uničenje in obveščanje o spremembah primerkov stanja (angl. state instance), ki pripadajo storitvi. Poleg tega definirajo tudi razširitev opisnega jezika spletnih

storitev (WSDL), ki tako razširjen omogoča eksplicitno deklaracijo javno dostopnih podatkov o stanju, ki pripada storitvi. Natančnejši opis teh mehanizmov je podan v literaturi [11].

2.2 Programska oprema Globus Toolkit

Programska oprema Globus Toolkit četrte generacije (GT4) je trenutno vodilna vmesna programska oprema (angl. middleware), namenjena graditvi ogrodnih sistemov. Je odprtokodne narave in sledi specifikacijam arhitekture OGSA. Daje nabor osnovnih ogrodnih storitev, ki jih lahko uporabimo pri graditvi ogrodne aplikacije, in nabor varnostnih mehanizmov s skupnim imenom ogrodna varnostna infrastruktura (Grid Security Infrastructure - GSI). Nekatere od teh storitev in mehanizmov, ki jih lahko uporabimo brez dodatnega programiranja, smo vključili tudi v razvito storitev posredovanja virov.

Poleg nabora že razvitih komponent, GT4 ponuja tudi razvojno okolje, namenjeno razvoju storitev po specifikacijah WSRF. Pomeni namreč implementacijo specifikacij WSRF v obliki programskih knjižnic, poleg tega pa tudi vso tehnologijo, potrebno pri razvoju in nameščanju storitev WSRF v izvajalno okolje. Izvajalno okolje orodjarne GT4 zagotavlja vse strežniške tehnologije, potrebne za izvajanje ogrodnih storitev (podpora JNDI, interpretor sporočil SOAP, aplikacijski strežnik itd.).

Razvojno okolje GT4 razvijalcem ponuja implementacijo specifikacij WSRF za programske jezike Java (Java Web Services Core), C (C Web Services Core) in Phyton (pyGridWare), vendar je Javanska implementacija najpopolnejša. Tudi sicer je orodjarna GT4 najbolj prilagojena razvoju storitev v programskem jeziku Java, poleg tega pa je večina že razvitih storitev, ki jih ponuja GT4, napisanih prav v Javi.

3 Zahteve pri posredovanju virov

Cilj projekta DataMiningGrid je bil razvoj ogrodnega okolja, ki omogoča souporabo porazdeljenih računskih in podatkovnih virov za podatkovno rudarjenje [1]. Odjemalci sistemu v obliki opisnega dokumenta predajo formaliziran opis več uporabniških poslov, sistem pa posle optimalno razvrsti na porazdeljene vire, jih izvede, rezultate pa shrani na dogovorjenem mestu. V nadaljevanju bodo podane predvidene uporabniške in sistemske zahteve do razvitega posrednika virov, ki skrbi za razvrščanje poslov.

3.1 Arhitekturne in tehnološke zahteve

Uporabljena arhitektura in tehnologija igrata pomembno vlogo pri integraciji z drugimi komponentami sistema. Odločili smo se za implementacijo v obliki storitve, kar omogoča fleksibilno nadgradnjo razvrščevalne politike posredovanja virov. Zahtevana je bila sposobnost razvrščanja in izvajanja poslov na računskih virih, ki pripadajo različnim administracijskim domenam, torej je bila potrebna integracija varnostnih mehanizmov. Zahtevano je bilo tudi sledenje specifikacijam WSRF, kar omogoča preprosto povezljivost z drugimi ogrodnimi storitvami.

3.2 Zahteve uporabniških poslov

Pri vzporednem izvajanju lahko med posameznimi uporabniškimi posli nastopajo različne vrste medsebojnih odvisnosti. V okviru projekta DataMiningGrid je bilo zahtevano le vzporedno izvajanje medsebojno popolnoma neodvisnih poslov.

Za podajanje velikega števila poslov pri parametričnih študijah je bila v okviru projekta razvita posebna opisna shema. Na njeni podlagi je mogoč formaliziran opis več uporabniških poslov v enem samem opisnem dokumentu. Pri tem posamezni posli pomenijo izvajanje istega izvršljivega računalniškega programa, vendar z različnimi parametri in/ali vhodnimi podatki. Opisni dokumenti, ki ustrezajo razviti shemi, so torej vhodni podatek posrednika virov, ki na njihovi podlagi ustvari posamezne uporabniške posle ter jih razvrstiti na ustrezne vire v ogrodju.

3.3 Politika razvrščanja

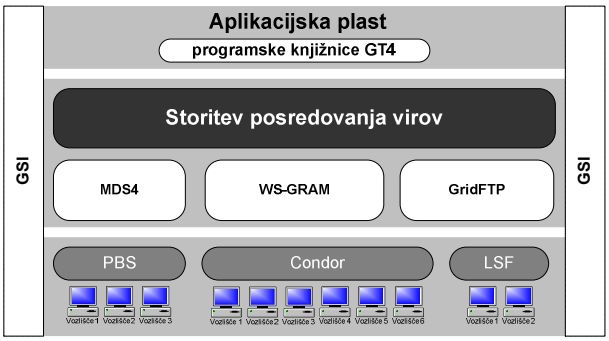
Politika razvrščanja določa, po kakšnih pravilih se iz seznama poslu ustrezajočih virov izbere vir, na katerem se bo posel dejansko izvedel. Politika razvrščanja je lahko preprosta, pri čemer se posel razvrsti na prvi vir iz seznama, lahko pa je kompleksnejša in pri razvrščanju upošteva tudi informacije, kot so trenutna obremenjenost virov, prioriteta poslov, cena izvajanja itd. V našem primeru smo se odločili za preprosto politiko razvrščanja, ki je težila k izbiri računalniške gruče, katere število trenutno prostih računskih virov presega število poslov, podanih v opisnem dokumentu. Tako smo minimizirali število podatkovnih prenosov, ki so potrebni pred izvajanjem poslov. Natančnejši opis algoritma razvrščanja je opisan v literaturi [12].

4 Storitev posredovanja virov

V tem poglavju bo podana umestitev storitve posredovanja virov v arhitekturo sistema in predstavljen potek implementacije storitve. Opisani bodo uporabljeni mehanizmi za zagotavljanje varnosti in nazadnje tudi diagram poteka klica razvite storitve.

4.1 Arhitektura

Arhitektura razvitega sistema je predstavljena na sliki 2. Na najvišji plasti arhitekture se nahajajo odjemalci, ki želijo varen dostop do računskih virov, katere predstavljajo različne računalniške gruče. Odjemalci v ta namen pripravijo opis poslov in ga predajo storitvi posredovanja virov, ki poskrbi za njihovo razvrščanje in izvedbo.



Slika 2: Umestitev razvite storitve posredovanja virov v arhitekturo sistema
Figure 2: Role of the resource-broker service in the overall system architecture.

Pri razvrščanju storitev posredovanja virov sodeluje tudi s storitvami MDS4, WS-GRAM in GridFTP vmesne programske opreme GT4. Za uspešno razvrščanje je namreč potrebna informacija o trenutnih zmožnostih in razpoložljivosti virov, ki jo daje storitev MDS4. Po izbiri računalniških gruč, na katerih naj bi se posamezni posli izvedli, storitev posredovanja virov posle preda storitvam WS-GRAM, ki so vmesnik do izbranih računalniških gruč. Le-te posle predajo nadzornikom lokalnih računalniških gruč (PBS, Condor, LSF) ter sprožijo in spremljajo potek njihovega izvajanja. Za pripravo izvajalnega okolja je pred samim izvajanjem potrebna še storitev za prenos podatkov GridFTP. Varnostne mehanizme zagotavlja ogrodna varnostna infrastruktura GSI, ki je prav tako del razvojnega okolja GT4. Podrobnejši opis arhitekture je podan v članku [13].

4.2 Implementacija storitve

Skladno z zahtevami, ki smo jih zastavili komponenti posredovanja virov, smo komponento po specifikacijah WSRF implementirali in jo namestili v vsebnik okolja GT4. Razvita komponenta nastopa v obliki storitve in je sposobna avtomatske izdelave seznama poslu ustreznih virov, razporejanja poslov na vire in avtomatskega združevanja rezultatov na predpisanem mestu v ogrodju. Kot opisni jezik poslov smo uporabili lastno rešitev, omenjeno v poglavju 3.2. Za zagotavljanje varnosti storitev uporablja mehanizme varnostne infrastrukture GSI. Med razvojem storitve smo težili k ponovni uporabi že obstoječe programske kode. Odločili smo se za preoblikovanje posrednika virov GridBus [7].

Med razvojem storitve smo uporabili programske knjižnice Java WS Core, ki pomenijo implementacijo specifikacij WSRF v Javi. Pri tej implementaciji stanja storitve nastopajo v obliki od storitve popolnoma ločenih programskih objektov, do katerih lahko storitev dostopa in nad njimi izvaja operacije. V našem primeru je posamezen primerek stanja pomenil informacijo o trenutnem stanju izvajanja uporabniških poslov. Knjižnice Java WS Core omogočajo različne možnosti shranjevanja stanja (npr. v podatkovni zbirki, datoteki itd.), v našem primeru smo se odločili za shranjevanje v obliki programskih objektov. Posamezen primerek stanja je tako programski objekt, ki hrani informacijo o trenutnem stanju in poteku izvajanja (čas začetka izvajanja, tranjanje, čas konca itd.) vseh poslov, ki so bili podani ob klicu storitve. Ob vsakem klicu storitve se torej ustvari nov primerek stanja.

Implementacija Java WS Core tedaj, ko storitvi pripada več primerkov stanja, predvideva implementacijski pristop tovarne primerkov (angl. factory/instance). Ta pristop zahteva dve ločeni storitvi. Prva je namenjena ustvarjanju primerkov stanja (angl. Factory Service), medtem ko je druga (angl. Instance Service) namenjena implementaciji same funkcionalnosti storitve. V programskem jeziku Java smo torej implementirali storitvi z imeni *RBFactoryService* in *RBInstanceService*. Implementacija storitve *RBFactoryService* je preprosta. Pomeni le implementacijo metode *createResource()*, ki je namenjena ustvarjanju primerkov stanja. Implementacija storite *RBInstanceService* je kompleksnejša in vsebuje dve pomembnejši metodi. S klicem prve odjemalec začne razvrščanje poslov na vire. Ta metoda je glavnina funkcionalnosti. Druga metoda omogoča odjemalcem dostop do poljubnega primerka stanja, ki pripadajo storitvi *RBInstanceService*. Za obe storitvi je treba napisati datoteki WSDL, ki se nekoliko razlikujeta od datotek WSDL za opis navadnih spletnih storitev.

V zvezi z ohranjanjem stanja je treba pri uporabi implementacije Java WS Core implementirati še dva razreda. Razred *ResourceBrokerResource* pomeni stanje storitve. Razred *ResourceBrokerResourceHome* je namenjen upravljanju vseh primerkov stanja, ki pripadajo storitvi *RBInstanceService*. Prek tega razreda lahko storitev *RBFactoryService* ustvari nov primerek stanja, storitev *RBInstanceService* pa lahko na podlagi naslova primerka stanja dostopa do točno določenega primerka, mu spreminja vrednosti lastnosti ali pa določen primerek stanja uniči. Naslavljanje primerkov stanja poteka na podlagi specifikacij WS-Addressing.

Ko so vsi štirje zahtevani razredi, ki pomenijo storitev WSRF implementirani in datoteki WSDL pripravljeni, je treba pripraviti še navodila za nameščanje storitve v strežniški vsebnik. Ta navodila vsebujeta opisna dokumenta WSDD (Web Service Deployment Descriptor) in JNDI (Java Naming and Directory Interface). S pomočjo orodja Ant se nato izdelajo nastavni razredi (angl. stub classes) odjemalca in storitve in storitev se namesti v strežniški vsebnik izvajalnega okolja GT4.

4.3 Zagotavljane varnosti

Varnostne mehanizme v razvitem sistemu zagotavlja infrastruktura GSI. Ta je zasnovana na šifriranju z javnim ključem, zagotavlja pa zasebnost (na ravni transporta in sporočil), integriteto sporočil, mehanizme

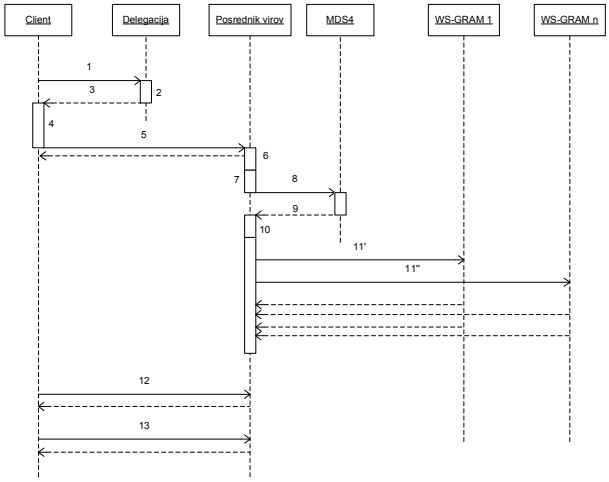
za avtentikacijo in avtorizacijo uporabnikov in mehanizme, ki omogočajo funkcionalnost enkratnega prijavljanja.

Integracija mehanizmov za zagotavljanje zasebnosti, integritete, avtentikacije in avtorizacije na samo implementacijo storitve ne vpliva, saj so zanjo potrebne le ustrezne nastavitve pri izdelavi nastavnih razredov in nameščanju storitve. Pri implementaciji storitve je bilo treba z varnostnega stališča upoštevati le delegiranje uporabniških poverilnic, ki omogoča enkratno prijavljanje. Ta je v našem primeru potrebna, saj storitev posredovanja virov med delovanjem kliče tudi druge storitve, ki prav tako zahtevajo avtentikacijo in avtorizacijo kličočega. Delegiranje uporabniških poverilnic poteka s pomočjo storitve delegiranja poverilnic (angl. Delegation Service), ki je del infrastrukture GSI in jo zagotavlja vmesna programska oprema GT4.

4.4 Diagram poteka klica razvite storitve

Poenostavljeni diagram poteka klica razvite storitve posredovanja virov je predstavljen na sliki 3. Vsaka od storitev, predstavljenih v diagramu, je dejansko implementirana kot dve ločeni storitvi, saj tako veleva programski model Java WS Core.

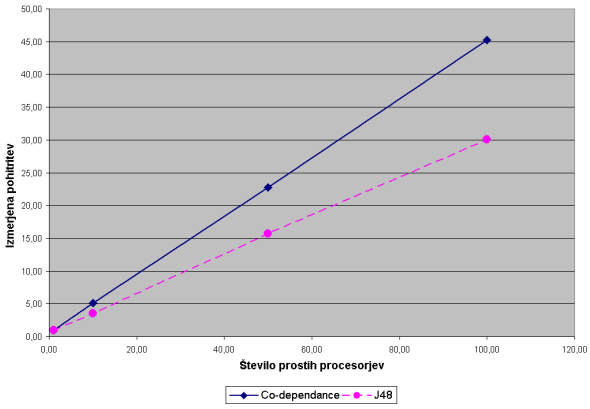
Klic storitve posrednika virov se začne z delegiranjem začasne varnostne poverilnice (angl. proxy certificate) storitvi delegiranja poverilnic (koraki 1,2 in 3). Nato odjemalec sestavi opisni dokument poslov (korak 4), pokliče storitev posredovanja virov in ji poda naslov delegirane poverilnice in opisni dokument poslov (korak 5). Storitev nato iz podanega naslova ustvari objekt, ki pomeni delegirano poverilnico, s pomočjo katere se storitev avtenticira in avtorizira ob klicu drugih storitev (korak 6). Nato na podlagi opisnega dokumenta ustvari opise posameznih poslov (korak 7). V naslednjem koraku storitev posredovanja virov kliče storitev MDS4. Od nje pridobi informacijo o razpoložljivosti in trenutnem stanju virov v ogrodju (koraka 8 in 9), nato pa na podlagi zahtev poslov sestavi seznam njim ustrezajočih virov. V koraku 10 določi vire, na katerih se bodo posamezni posli izvedli. Posamezne opise poslov nato preda izbranim storitvam WS-GRAM (korak 11). Posamezna storitev WS-GRAM pripravi izvajalno okolje (pri tem uporabi storitev GridFTP), nato pa prejeti posel preda v izvajanje lokalni računalniški gruči. Storitev WS-GRAM tudi spremlja izvajanje posameznih poslov ter o dogodkih obvešča storitev posredovanja virov. Le-ta glede na dogodke spreminja vrednosti lastnosti ustreznega primerka stanja, ki vsebuje informacijo o izvajanju uporabniških poslov. Ko se vsi uporabniku pripadajoči posli končajo (korak 12), odjemalec poskrbi za uničenje primerka stanja, ki pripada tem poslom (korak 13).



Slika 3: Diagram poteka klica storitve posredovanja virov
Figure 3: Sequence diagram of the resource-broker service invocation.

5 Rezultati

Razvito storitev smo namestili v testno ogrodno okolje, ki se je razprostiralo prek treh evropskih držav, in sicer Velike Britanije, Nemčije in Slovenije, ter jo preizkusili. Izmerili smo potek pohitritve (angl. speed-up) v odvisnosti od števila prostih računalniških procesorjev, na katere storitev posredovanja virov razvrsti po 100 uporabniških poslov.



Slika 4: Izmerjena poteka pohitritev
Figure 4: Measured speed-up values.

Ugotovili smo, da je pri izvajanju časovno zahtevnejših poslov, ki jih pomeni vzporedno izvajanje stotih instanc evolucijskega algoritma za simulacijo regulatornih genetskih mrež (Co-dependance algoritem), pohitritev s številom prostih procesorjev naraščala približno po premici s smernim koeficientom ½ (Slika 4). Pri časovno manj zahtevnih poslih, ki jih pomeni vzporedno izvajanje stotih instanc klasifikatorja na podlagi odločitvenih dreves (algoritem J48), je bil ta koeficient približno enak 1/3 (sSlika 4).

Ocenili smo tudi skalabilnost (angl. scale-up) sistema, ki se je izkazala kot zelo dobra. Podrobnejši

opis meritev in dobljenih rezultatov smo objavili v člankih [14] in [15]. Rezultati so pokazali tudi določene pomanjkljivosti razvite storitve. Pri velikemu številu relativno kratkih poslov se je na primer pokazal problem ne optimalnega razvrščanja poslov, ki je bistveno podaljšal skupni čas izvajanja poslov podanih v eni uporabniški zahtevi [15].

6 Sklep

V pričujočem delu smo opisali metodologijo razvoja storitve posredovanja virov po specifikacijah WSRF BP 1.0. Za lasten razvoj storitve smo se odločili, ker sorodne programske komponente, ki bi ustrezala prav vsem zastavljenim zahtevam, ni bilo. Pri implementaciji smo uporabili razvojno okolje GT4, ki je trenutno de-facto standardna programska oprema za razvoj ogrodnih aplikacij.

Razvita storitev je bila nameščena v realno ogrodno okolje in preizkušena. Med preizkušanjem so se pokazale določene pomanjkljivosti, katerih odpravljanje je naslednji korak v razvoju sistema. Kot nadaljnje delo bi tako lahko navedli izboljšanje razvrščevalne politike razvite storitve. Poleg tega bi pri razvoju storitve lahko uporabili več zmožnosti, ki nam jih ponujajo specifikacije iz družine WSRF BP 1.0. Po specifikacijah WS-ResourceLifetime (del specifikacij WSRF) bi namreč lahko implementirali stanje, ki ga odjemalcu ni treba eksplicitno uničiti, temveč se uniči samo, ko mu poteče življenjska doba. Tako bi rešili problem naraščanja števila primerkov stanja, ki jih odjemalcem zaradi različnih vzrokov (npr. zaradi odpovedi omrežja) ni uspelo uničiti. Nadaljnje izboljšave bi dosegli z uporabo specifikacij WS-Notification in tako zagotovili obveščanje odjemalcev ob spremembah vrednosti lastnosti stanja. S tem bi razbremenili strežnik, saj periodično povpraševanje po vrednostnih stanja ne bi bilo več potrebno.

Zahvala: Raziskave, opisane v tem delu so bile delno opravljene v okviru evropskega projekta šestega okvirnega programa DataMiningGrid, številka pogodbe 004475.

7 Literatura

[1] Stankovski V., Trnkoczy J., Swain M., Dubitzky W., Kravtsov V., Schuster A., Niessen T., Wegener D., May M., Rohm M., Franke J., Digging deep into the data mine with DataMiningGrid, IEEE internet computing, 2008, letn. 12, št. 6, str. 69-76.

[2] Kishimoto H., Jordan C., GFD-I.123: Defining the Grid: A Roadmap for OGSA Standards Version 1.1, 2008, dostopno na http://www.ogf.org/, zadnji dostop 18. nov. 2008.

[3] Uradna spletna stran organizacije Open Grid Forum (OGF), http://www.ogf.org , zadnji dostop 18. nov. 2008.

[4] Foster I., Maguire T., Snelling D., GFD-R-P.072: OGSA WSRF Basic Profile 1.0, 2006, dostopno na http://www.ogf.org/, zadnji dostop 18. nov. 2008.

[5] Uradna spletna stran projekta Globus Toolkit, http://www.globus.org/, zadnji dostop 18. nov. 2008.

[6] Foster I., What is the Grid? A three-point checklist, Grid Today 2002, dostopno na http://www-fp.mcs.anl.gov, zadnji dostop 18. nov. 2008.

[7] Venugopal S., Buyya R., Winton L., A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids, Technical Report, GRIDS-TR-2004-1, University of Melbourne, 2004.

[8] Huedo E., Montero R. S., Llorente I. M., A framework for adaptive execution in grids, Software Practice & Experience, 2004, letn. 34, št. 7, str. 631-651.

[9] Abramson D., Buuya R., Giddy J., A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker",Future Generation Computer Systems, 2002, letn. 18, št. 8.

[10] Allen G. et al., Enabling applications on the grid: a Gridlab overview. Int'l Journal of High-Performance Computing Applications, 2003, letn. 17, št. 4, str. 449-466.

[11] Foster I., Kishimoto H., Savva A., Berry D., Djaoui A., Grimshaw A., Horn B., Maciel F., Siebenlist F., Subramaniam R., Treadwell J., Reich J. V., GDF-I 080-The Open Grid Services Architecture, Version 1.5, dostopno na http://www.ogf.org/documents/GFD.80.pdf, zadnji dostop 18. nov. 2008.

[12] Kravtsov V., Niessen T., Stankovski V., Schuster A., Service-based Resource Brokering for Grid-based Data Mining, In Proceedings of The International Conference on Grid Computing and Applications (GCA'06), Las-Vegas, USA, 2006.

[13] Stankovski V., Swain M., Kravtsov V., Niessen T., Wegener D., Kindermann J., Dubitzky W., Grid-enabling data mining applications with DataMiningGrid: An architectural perspective, Future Generation Computer Systems Journal, 2008, letn. 24 , št. 4, str. 259-279.

[14] Kravtsov V., Niessen T., Stankovski V., Schuster A., Dubitzky W., Grid Resource Management for Data Mining Applications, Grid Technologies for Knowledge-Based Industries and Businesses Workshop, IST Event, Helsinki, 2006.

[15] Trnkoczy J., Stankovski V., Improving the performance of Federated Digital Library services, Future Generation Computer Systems, 2008, letn. 24, št. 8, str. 824-832.

Jernej Trnkoczy je diplomiral leta 2003 in magistriral leta 2007 na Fakulteti za elektrotehniko v Ljubljani. Sprva se je kot raziskovalec zaposlil na Katedri za gradbeno informatiko na Fakulteti za gradbeništvo in geodezijo, v letu 2007 pa se je pridružil Laboratoriju za digitalno obdelavo signalov, slik in videa na Fakulteti za elektrotehniko. Njegovo raziskovalno delo obsega raziskave na področju porazdeljenih sistemov, zlasti tehnologij ogrodja in tehnologij vsak z vsakim.