Nemzeti Információs Infrastruktúra Fejlesztési Intézet

Grid szolgáltatások beágyazott absztrakt protokollokkal

(tanulmány és rendszertechnikai terv)

Készítette
Stefán Péter
Szalai Ferenc
Vitéz Gábor

Magyar Informatikai Erőforráshálózat (Grid közmű) alapjai (MEGA) projekt
(NKFP OM-00262,00263,00264,00266,00268/2004)

NIIF Intézet, Budapest
2006. június 30.

Tartalomjegyzék

1. Bevezetés

A grid rendszerek együttműködésének vizsgálata során nagyon fontos meghatározni azt, hogy a jelenleg széles körben elterjedt grid köztesréteg megoldások milyen szolgáltatásokat használnak, azokat milyen felületen keresztül nyújtják, és az adott felületen milyen módon, milyen struktúrában cserélnek információt. Mindez azért lényeges, mert ha a különböző grid rendszerek együttműködését viszgáljuk, akkor tisztában kell lennünk a jelenleg használatos megoldásokkal, trendekkel, felhasznált kommunikációs struktúrákkal, protokollokkal és ennek alapján tudunk javaslatot tenni arra, hogy melyek azok a szolgáltatások, amelyek minden grid rendszerben megvannak, és ezek, az adott grid filozófián nem sértve, hogyan illeszthetők egymáshoz.

E tanulmány legfontosabb célkitűzése tehát, hogy bemutassa a ma népszerűen használt grid köztesrétegeket, a szolgáltatások szemszögéből, és felvázoljon egy olyan szolgáltatási modellt, amely képessé teheti a mi rendszerünket az azokkal való kommunikációra.

A tanulmány első része bemutatja a Globus ToolKit 2-4, a JINI grid, a LHC Grid, a NorduGrid ARC, valamint a web-szolgáltatás alapú szoftverek legfontosabb jellemzőit. A 7. fejezet bemutatja azt, hogy hogyan lehet a külső, "idegen" grid rendszerek és a Grid Közmű keretén belül fejlesztett rendszer közös szolgáltatási felületét elkészíteni, illetve a megfelelő infrastrukturális elemet kialakítani (gateway architektúra). A 8. fejezet foglalkozik az információs rendszerek információ-aggregáció megoldásaival, a 9. fejezet pedig a felhasznált beágyazott absztrakt protokollt írja le.

2. A Globus szolgáltatásai

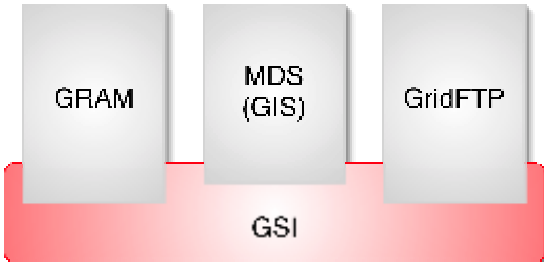
A Globus Toolkit (GT) szoftver az egyik legrégebben használt és ezért legelterjedtebb grid köztesréteg [6]. Mára két főbb változatát használják: a 2-es sorozatot, amely egyedi TCP/IP-alapú megoldásokat nyújt, illetve a 4-es sorozatot, amely már web szolgáltatás interfésszel rendelkezik. A GT, mint neve is mutatja, elsősorban egy eszköztár, amellyel tetszőleges, testreszabott grid rendszerek is felépíthetők. Emiatt nagyon népszerű nemcsak, mint szoftver-gyűjtemény, hanem mint API is. A Globus, azon túl, hogy önállóan is használják, alapját képzi más grid középrétegeknek, mint például az LCG, vagy az ARC szoftvereknek is. A GT szolgáltatásai, amelyek leírására kitérünk, az alábbiak:

- Biztonsági rendszer (Grid Security Infrastructure, GSI),
- Globus Erőforrás Menedzser (Globus Resource Access Manager, GRAM),
- Grid Információs Rendszer (Metacomputing Directory Service, MDS),
- Adat-átviteli rendszer (GridFTP).

2.1. Biztonsági rendszer (GSI)

A GT biztonsági rendszere alapvetően állapottartó autentikált kapcsolatokon alapuló, X.509 tanúsítványokat használó rendszer, amely a teljes Globus rendszert átfogja (2.1. ábra), azaz minden szolgáltatás ugyanazt a biztonsági modellt használja, statikus módon, a rendszerbe alkalmazás-könyvtár szinten bekódolva. A megoldás azért állapottartó, mert a szolgáltatások igénybevevője, illetve biztosítója egyaránt, állapotot őriz a kölcsönös azonosításuk állapotáról. Az állapot nem más, mint egy X.509 tanúsítvány segítségével aláírt úgynevezett proxy tanúsítvány, amelyet a felhasználó, az

általa igényelt szolgáltatáshoz való hozzáférés folyamán felmutat, és attól függően, hogy az igényelt szolgáltatás elfogadja-e azt a tanúsítvány-hatóságot, amely aláírta a felhasználó eredeti tanúsítványát, engedi vagy nem engedi hozzáférni a felhazsnálót a szolgáltatásokhoz. Fontos eleme, egyben limitációja a rendszernek az ideiglenesen létrehozott tanúsítvány, amely általában egy felhasználói ülés (user session) időtartamára, de legfeljebb 12 órán keresztül létezik. Ezt a felhasználó egy újabb ülés során meghosszabbíthatja. A GSI specifikációja a RFC3820 [24] dokumentumban található.



2.1. ábra: A GT 2 sematikus biztonsági architektúrája.

A biztonságnak fontos eleme és követelménye, hogy a felhasználó X.509 tanúsítványának privát kulcsa nem kerülhet illetéktelen kezekbe, illetve ennek bekövetkezése esetén a felhasználó jogosultságai kompromittálódhatnak, mivel a GT implicit szabálya, hogy két, ugyanazon tanúsítvánnyal aláírt ideiglenes proxy tanúsítványt felmutató entitás automatikusan megbízik egymásban.

Az X.509 kulcspár a felhasználói autentikáción túlmenően felhasználható még a későbbiek során a kommunikáció titkosítására is.

A jogosulttság-kezelés (authorizáció) egy speciális állomány a grid-mapfile segítségével töténik: a file leképzi a felhasználó tanúsítványában szereplő megkülönböztető neveket (Distinguished Name, DN) helyi operációs rendszer szintű jogosultságokra.

Filozófiában teljesen más megoldást használnak a GT 3-nál magasabb számú verziói. Nagyon fontos különbség az eddigiekhez képest, hogy maga a keretrendszer, amelyben a szolgáltatásokat értelmezzük is teljesen más, mint a 2.x széria esetén volt: a rendszer elsősorban web szolgáltatásokra, azaz elemi, illetve összetett szolgáltatásokból álló alapelemekre épül. Egy web szolgáltatás általában egy meghatározott hálózati protokollon elérhető függvény-gyűjtemény, amelyet hálózati web protokollon lehet elérni, és amelyhez a hozzáférés, illetve az entitás viselkedése szabályozható.

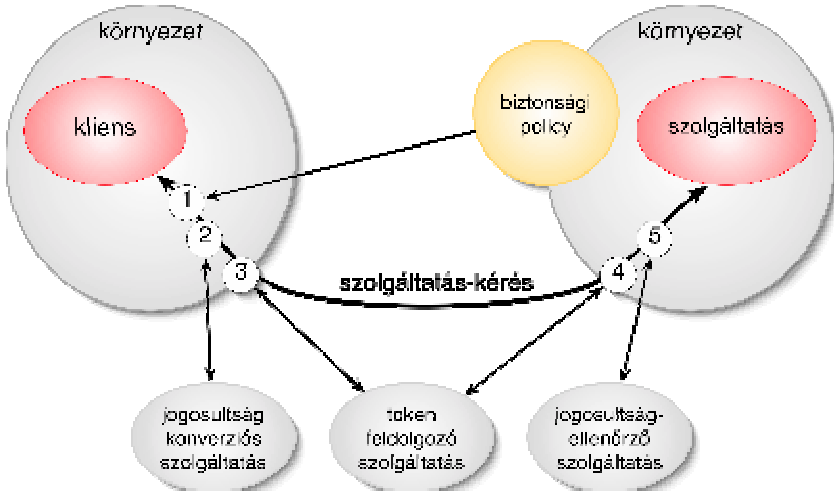
A számos web szolgáltatás közül az egyik építőelem a biztonsági infrastruktúra szolgáltatás, amely leginkább egy biztonsági modulhoz hasonlít, az alábbi feladatokat végezve:

- felhasználó autentikáció, felhasználó identitásának ellenőrzése,
- felhasználó authorizáció, azaz a felhasználó jogosultságainak, illetve az elérni kívánt szolgáltatás jogosultságainak vizsgálatával annak eldöntése, hogy a felhasználó hozzáférhet-e az adott szolgáltatáshoz,
- jogosultság-reprezentáció konvertálás, azaz a jogosultsági adatok egyik megjelenési formájából, egy másik, ekvivalens, megjelenési formába való átalakítása (X.509-ből Kerberos ticket),
- identitás konverzió, azaz a felhasználó identitásának átalakítása egyik tartományból a másik tartományba, illetve
- auditálás, azaz a biztonsági események naplózása.

Egy tetszőleges szolgáltatás, amely szeretné eldönteni, hogy egy kliens jogosult-e az őt elérni, nem maga határozza meg ezt a döntést, hanem megkérdi a biztonsági szolgáltatást, akiben megbízik, és aki erre, a felhasználó intézményének védelmi stratégiájába illeszkedő, konzisztens választ tud adni. E megközelítésnek két fontos előnye is van: egyrészt a szolgáltatások funkciói letisztultabbak, másrészt a biztonság testre szabható, azaz létezhet, akár egyidőben, olyan biztonsági szolgáltatás egyazon intézményen belül, amely megengedőbb, és amelyik restriktívebb. A "biztonsági szint" finom-hangolásához csakis a biztonsági szolgáltatást kell átalakítani, az őt igénybevevő egyéb szolgáltatásokat nem.

Nagyon fontos, hogy a szolgáltatások megfelelő felülettel rendelkezzenek. A felület mögött az implementáció változhat, de a felület állandósága, illetve szabványossága biztosítja az egységes közös nyelvet a szolgáltatások között. A GT 3-as sorozat biztonsági architektúráját a 2.2. ábra mutatja. A modell működése az alábbi:

1. A kliens letölti a szolgáltatás biztonsági stratégiáját. (Minden szolgáltatás megmondhatja magáról, hogy milyen szolgáltatási, biztonsági szintet követel meg az igénybevevőtől.)
2. Amennyiben szükséges, a GT a kliens oldalon átkonvertálja a jogosultság megjelenéseket a szolgáltatás által kért formátumba, egy külső konverziós szolgáltatás segítségével.
3. A kliens egy külső, biztonsági token feldolgozási szolgáltatáshoz fordul avégett, hogy az az ő jogosultság információit feldolgozza. (Ez tehermentesíti az alkalmazást attól a biztonsági mechanizmusok részleteinek ismerete alól.)
4. Hasonlóan tesz a szolgáltatás is a kliens által biztosított jogosultsági információkkal: visszaellenőrzi a klienst. (Használhatják ugyanazt a szolgáltatást, de használhatnak különböző szolgáltatásokat is.)
5. A sikeres autentikációt követi a jogosultság-ellenőrzés, azaz a szolgáltatás, a kliens által biztosított jogosultsági információk alapján el kell dönteni, hogy az adott kliens hozzáférhet-e az adott szolgáltatáshoz.
6. Amennyiben igen, megkezdődik a szolgáltatás igénybevételével kapcsolatos üzenetváltás a kliens és a szolgáltatás között.



2.2. ábra: A GT 3 GSI architektúrája.

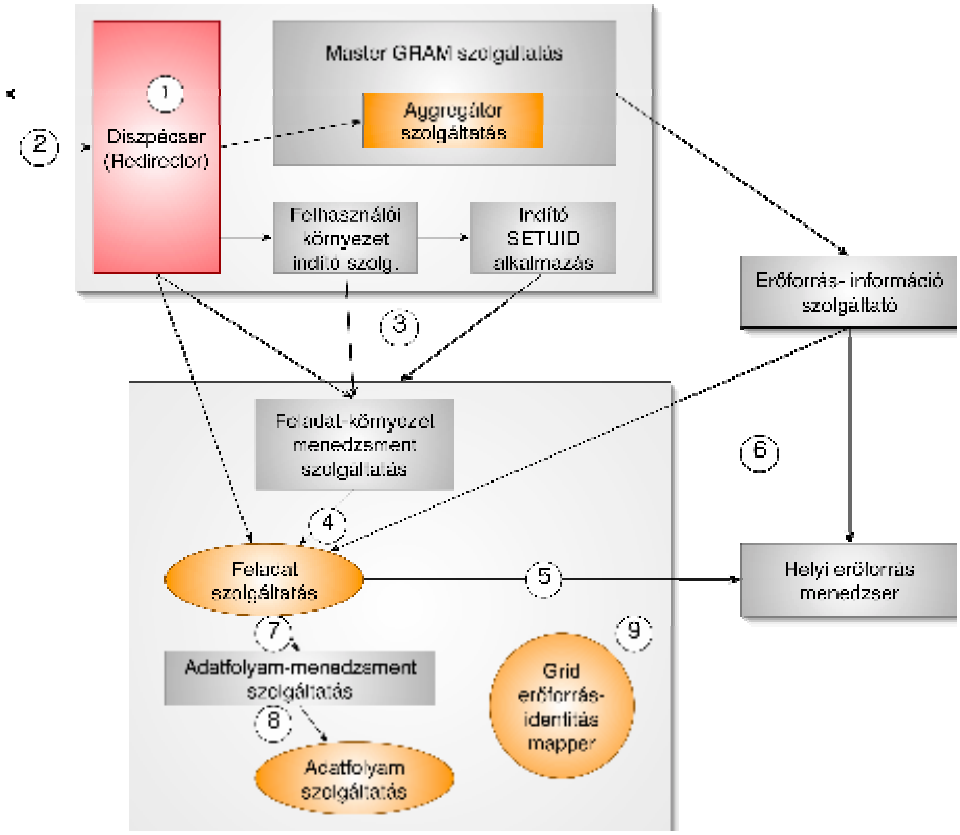
A GT 3-as sorozat biztonsági alrendszere képes állapotmentes, azaz korábban kölcsönös biztonsági környezet létrehozása nélkül kialakított kommunikációra is.

2.2. Erőforrás menedzser (GRAM)

A Globus Erőforrás Menedzser (Grid Resource Allocation Manager, GRAM) szolgáltatás legfontosabb feladata az erőforrás, illetve feladat koallokáció biztosítása. A GT GRAM szolgáltatás lehetővé teszi számításigényes felhasználói feladatok távoli futtatását, monitorozását, illetve menedzselését. Ez oly módon történik, hogy minden erőforrás rendelkezik egy erőforrás-bejárattal (gatekeeper), amely egyben interfész a lokális erőforrás menedzser (Local Resource Management System, LRMS) felé. Minden a grid irányából érkező (azaz nem lokális) felhasználó előbb az erőforrás információs rendszerét kérdezi meg az erőforrás állapotáról, és ettől az információtól függően átadja a bejáratnak a számítási feladatát. A bejárat, a GSI modulban részletezett módon, autentikálja a felhasználót és átveszi a feladatot. A feladat részét képzi egy feladat leírás, amelyet valamilyen magas szintű nyelven fogalmaznak meg. A GT rendszerek esetén ez a nyelv az erőforrás-leíró nyelv (Resource Specification Language, RSL). A bejárat szolgáltatás a feladat átvétele során az alábbiakat végzi:

- foglalási igényt jelez a feladat követelményeinek megfelelő erőforrásokra,
- regisztrálja a feladatot, illetve a lefoglalt erőforrások státuszát az információs rendszerben,
- a kérelmező jogosultságait statikusan leképzi helyi felhasználói jogosultságokra, a felhasználó tanúsítványának DN attribútuma alapján (grid-mapfile),
- speciális felügyelőfolyamatot indít, amely a feladatot annak teljes futási ideje alatt végig figyelemmel kíséri,
- átadja a feladatot egy helyi erőforrás menedzsernek (LRMS), vagy közvetlenül az erőforrás operációs rendszerének.

Magasabb verziószámú, web-szolgáltatás alapú GT rendszerek esetén a GRAM szolgáltatás egymásra épülő, elemi web szolgáltatásokból áll, mint azt a 2.3. ábra illusztrálja.



2.3. ábra: GRAM alrendszer felépítése elemi web szolgáltatásokból GT 3 esetén.

A GRAM szolgáltatás működése röviden az alábbi:

1. A GRAM lelke a Master szolgáltatás, amely koordinálja az erőforrás-kérés, allokáció, és monitorozási tevékenységeket. Alapvetően gondoskodik arról, hogy:
   a. a GRAM szolgáltatás látható legyen a külvilág számára,
   b. direkt kommunikáció jöjjön létre a feladat környezete, illetve a diszpécser (illetve a külvilág) között,
   c. az aggregátor szolgáltatást használva az erőforrás aktuális állapotát leíró információ jusson el a külvilágba,
   d. esetleges erőforrás regisztrációt végezzen egy külső adatbázisba, index szerverbe.
2. A kliens egy feladat futtatásának szándékával egy speciális createService szolgáltatás igénybevételével fordul a GRAM diszpécser szolgáltatásához, amely a kérést fogadja, és továbbítja egy másik szolgáltatás felé, attól függően, hogy a felhasználói kérésnek már van-e létező, korábban létrehozott folyamata, vagy nincs.
3. Amennyiben a felhasználónak még nincs lekezelt kérése, akkor, kölcsönös autentikáció, illetve authozizáció után, a master szolgáltatás a grid-mapfile alapján, a feladat-környezet menedzsment szolgáltatás igénybevételével, létrehoz egy felhasználói környezetet (egy új operációs rendszer folyamat), a grid-ből jövő jogosultságokat helyi felhasználói jogosultságokra leképezve. Miután létrejött e környezet (vagy az már korábban létezett) a diszpécser továbbítja a felhasználói kéréseket a feladat-környezetnek.
4. A feladat-menedzsment szolgáltatás elkészíti a feladat szolgáltatást. A feladat szolgáltatás egy konkrét feladatot helyi erőforráson ütemez, monitorozza, és eseményeket képes annak állapotáról küldeni.

5. A feladat szolgáltatás elhelyezi a feladatot a helyi erőforrás-menedzser rendszerben.
6. A helyi erőforrás-menedzser információt szolgáltat az erőforrás állapotáról.
7. Annak érdekében, hogy a futó feladat a szabványos kimenetét és hibakimenetét, mint folyam-adatot vissza tudja adni a kliensnek, a feladat szolgáltatás az adatfolyam menedzsment szolgáltatáshoz fordul…
8. … aki létrehoz egy-egy adatfolyam szolgáltatást, egyet a kimenetnek, és egyet a hibakimenetnek.
9. A Grid erőforrás-identitás leképző alrendszer, amely a felhasználói környezet része, biztosítja azt, hogy a kliens, illetve a feladat környezetben futó feladat, illetve a hozzá kapcsolódó szolgáltatások kölcsönösen autentikálni tudják egymást.

A GRAM a feladatok leírására RSL feladatleírót [9] használ, és a kliensekkel GRAM protokollt beszél, amely üzenet-formátumát a [10] dokumentum foglalja össze. (Magasabb verziószámú, alapvetően web szolgáltatás alapú Globus rendszerekben a GRAM protokollt továbbfejlesztették, lásd: [11] interfész leírást.)

2.3. Grid információs rendszer (MDS)

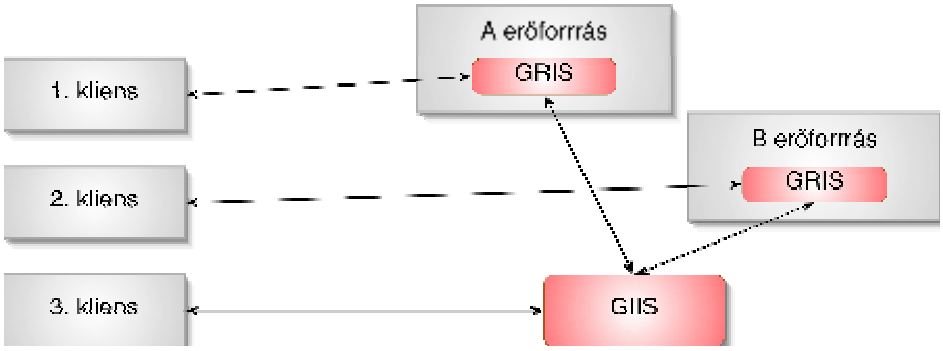
A grid információs rendszer (Metadata Discovery Service, MDS) legfontosabb feladata az, hogy a grid rendszerben használt entitások (erőforrások, feladatok) állapotáról viszonylag friss információt biztosítson, és tegyen elérhetővé a felhasználók számára. A GT 2 MDS rendszere alapvetően egy hierarchikus felépítésű LDAP szerver, amely egységes névteret biztosít, és amelyben minden szervezet rendelkezik egy saját al-névtérrel, amelybe erőforrásainak adatait regisztrálhatja. Az MDS alapvetően három komponensből áll:

- Mag-információ szolgáltatók (Core Information Providers, CIP),
- Grid Erőforrás Információs Rendszer (Grid Resource Information Service, GRIS),
- Grid Index Információs Szolgáltatás (Grid Information Indexing Service, GIIS).

A CIP alapvető feladata, hogy az operációs rendszer, illetve az erőforrás-menedzser specifikumait eltakarva erőforrás-információt juttasson az LDAP szerver GRIS komponensébe. Ezek általában script gyűjtemények, amelyeket a főbb operációs rendszer, LRMS disztribúciókhoz elkészítettek, de egy saját megoldáshoz is könnyen elő lehet őket állítani. Ezt a komponenst arra is rá lehet venni, hogy speciális adatokat is eljuttasson az MDS-be.

A GRIS fogadja, feldolgozza, és az LDAP névtérben tárolja az erőforrásról, annak állapotáról érkezett információt. A tárolás mind permanens tárolást, mind időleges tárolást (cache-elést) is biztosít a hatékonyabb keresés érdekében. A GRIS-t alapvetően arra tervezték, hogy bárki számára aktuális információt adjon a lokális erőforrás főbb paramétereiről. A GRIS, bár a kliens által önállóan is megszólítható szolgáltatás, általában regisztrálja önmagát, és az általa biztosított adatokat a GIIS katalógusba.

Egy GIIS csomópont általában egy szervezethez, szervezeti egységhez tartozik, és bejegyez több GRIS által küldött erőforrás-adatot, illetve azok elérhetőségeit. A GIIS aggregálja az erőforrás-információt, ezáltal hatékonyabb keresést tesz lehetővé. A GIIS, tervezési filozófiájából adódóan, független más GIIS csomópontoktól, ugyanakkor tárolhat információt más GIIS csomópontokról is. Így egy egész GIIS hierarchia építhető föl. A GT MDS sematikus felépítését, és működését a 2.4 ábra mutatja.



2.4. ábra: A GT információs rendszerének sematikus felépítése.

A web szolgáltatás alapú megoldások információs rendszere hasonlít a GT 2 szoftvernél kimunkált rendszerhez, sőt jelentősen tovább is fejleszti azt. A web-szolgáltatás alapú MDS alapvetően az "aggregációs keretrendszerre" épít, amelyen belül három fontosabb szolgáltatást is definiálnak:

- információs szolgáltatás, amelybe az egyes erőforrások regisztrálhatják magukat,
- trigger szolgáltatás, amely legfontosabb feladata az, ha az egyes erőforrások státusza megfelel egy bizonyos feltételrendszernek, akkor figyelmeztetést küld,
- archiváló szolgáltatás, amely az erőforrások múltbeli állapotáról őriz rekord-bejegyzéseket.

A Globus MDS rendszere az MDS LDAP sémát használja [1][7], kommunikációs protokollja LDAP. A normál MDS séma helyett a rendszerbe betölthető a GLUE 1.1 séma is [2]. A GT 3 rendszer natív módon használja a GLUE sémát, ugyanakkor az adatreprezentációja XML, kommunikációs protokollja HTTP fölött továbbított SOAP.

2.4. GridFTP szolgáltatás

A GridFTP szolgáltatás elsősorban nagy méretű adatállományok, különböző grid csomópontok közötti mozgatására szolgál. Technikai felépítésében annyival több a hagyományos FTP szolgáltatásnál, hogy lehetővé teszi a tanúsítvány alapú autentikációt mind a vezérlő, mint az adatátviteli csatornán. A felhasználó egy speciális GridFTP kliens segítségével fordul a GridFTP szolgáltatáshoz, ahol a klasszikus felhasználói név és jelszó megadása helyett a felhasználó a saját proxy tanúsítványával autentikálja magát, és az adott erőforrás hozzáférési-szabályozása (access policy, illetve grid-mapfile) alapján kap bebocsátást a szerverre, ahol az FTP szolgáltatásnál megszokott műveleteket végezheti.

A magasabb verziószámú GT rendszerek esetén a GridFTP szolgáltatás kiegészül még az alábbi elemekkel:

- párhuzamos IO támogatása céljából többszörözött adatcsatornák,
- részleges file átvitel lehetősége,
- autentikált adatcsatorna,
- moduláris felépítés, például HTTP fölötti IO lehetősége.

2.5. Ami hiányzik a Globus rendszerből

A GT, mint azt a neve is mutatja elsősorban egy keretrendszer, amely segítségével grid rendszerek építhetők fel, különböző filozófiák alapján. Így számos olyan funkció hiányzik még belőle, amely más, a Globus rendszerre építő köztesrétegekben megjelent:

- erőforrás bróker,
- mérőrendszer,
- monitorozó rendszer,
- számlázó rendszer,
- authorizációs alalrendszer,
- virtuális szervezetek (Virtual Organization, VO) kezelésére alkalmas alrendszerek.

3. A JINI Grid szolgáltatásai

A Java alapú, Jini technológia, amelyet a SUN fejlesztett ki, lehetővé teszi azt, hogy erőforrások, emberi beavatkozás nélkül, dinamikus módon ad-hoc hálózatba szerveződjenek [12]. Az erőforrások bármikor "bejelentkezhetnek" egy ún. regisztrációs szolgáltatásba, és bármikor kijelentkezhetnek onnan.

A Jini alapvetően egy speciális programozási paradigmára épít: a proxy és a backend szolgáltatások paradigmájára. E paradigma azt mondja, hogy egy szolgáltatás alapvetően két részből épül föl: egy frontend-ből (proxy), amely egy letölthető Java objektum, illetve egy backend részből, amely megvalósítja a szolgáltatás funkcióit. E két entitás fizikailag két külön erőforráson fut, ahol speciális alkalmazás-specifikus protokollal kommunikálhatnak egymással (3.1. ábra). Java nyelven írt programok esetén a Java távoli eljáráshívás (Remote Method Invocation, RMI) eljárását is használhatják kommunikációra.

3.1. ábra: A Jini alapkoncepciója.

A Jini alapon felépített grid rendszerek alapvetően az alábbi alapszolgáltatásokból, illetve kiegészítő szolgáltatásokból épülnek föl:

- Lookup szolgáltatás (Lookup Service, LUS),
- Bróker szolgáltatás (Broker Service, BS), illetve
- Compute szolgáltatás (Compute Service, CS).

3.1. A Lookup szolgáltatás (LUS)

A Lookup szolgáltatás alapvetően egy név regisztrációs szolgáltatás, amelyben objektumok, szolgáltatások regisztrálhatják magukat. A szolgáltatás adatsémáját a ServiceItem nevű Jini objektum adja. A regisztrált szolgáltatások elérése, illetve regisztrációja az alábbi protokoll alapján történik:

- A kliens a felderítési protokoll segítségével találja meg a LUS-t. A felderítés multicast IP csomagok segítségével történik.
- A szolgáltatások szintén e protokollt használják a LUS szolgáltatások felderítésére, majd

- a szoláltatások a ServiceItem objektum segítségével regisztrálják a felületüket (proxy) a LUS szolgáltatásban (join protokoll).
- A kliensek a lookup protokoll segítségével térképezhetik fel a LS-en elérhető szolgáltatásokat.
- A megfelelő szolgáltatás kiválasztása után annak ServiceItem objektumát a kliens megkapja. Nem maga a teljes szolgáltatás töltődik le, hanem annak csak az ún. proxy része, amely Jini terminológiában a szolgáltatás felülete amely saját belső protokoll segítségével kommunikálhat a szolgáltatást ténylegesen megvalósító elemeivel.

Mivel a Java támogatja a kód portabilitást, az objektum formájában visszaadott szolgáltatásnak nem kell a kliensen telepítve lennie, az mindig futási időben töltődik le a kliensre, ezáltal gondoskodva arról, hogy mindig a legfrissebb verzió kerüljön le.

A LUS a proxy objektumokat meghatározott ideig tárolja. Amennyiben ez az idő lejár a szolgáltatásnak újra kell önmagát regisztrálni. Ha ezt nem teszi meg, törlődik a LUS-ból.

A LUS szolgáltatásokkal történő kommunikációra az alábbi protokollokat használja:

- Multicast Request Protocol, LUS felderítésére,
- Multicast Announcement Protocol, LUS szolgáltatások hirdetésére,
- Unicast Discovery Protocol, a LUS szolgáltatással történő direkt kommunikációra [13].

3.2. Compute szolgáltatás (CS)

A Compute szolgáltatás (Compute Service, CS) legfontosabb feladata az, hogy lehetővé tegye azt, hogy egy adott erőforrást, mint Java virtuális gépet egy kliens számítási idő-igényes feladatok végrehajtására használja. A CS-t a kliens a LUS szolgáltatáson keresztül éri el, amely szolgáltatás biztosítja azt, hogy a kliensről adatot tudjunk átvinni az erőforrásra, majd azt, az erőforrás proxy-ját letöltve, távolról menedzselni tudjuk. A CS a kapott feladatot a JVM round robin ütemezőjét használva futtatja le. A felhasználók regisztrálhatnak egy ún. Listener szolgáltatásra (Listener Service, LS) is, amely aszinkron üzeneteket, így a feladat eredményeit képes továbbítani azok felé.

A CS alapesetben Java nyelven írt programokat képes futtatni, de más szolgáltatásokkal kiegészítve lehetőség van tetszőleges bináris végrehajtható állomány futtatására is. Java feladat esetén fontos, hogy a feladat implementálja a Jmtask interfészt, tehát magának az alkalmazásnak tudatában kell lennie arról, hogy Jini griden fut majd.)

3.3. A bróker szolgáltatás (BS)

A bróker szolgáltatás (Broker Service, BS) legfontosabb feladata, hogy minél több erőforrást térképezzen föl, és tegyen a kliensek által regisztrálhatóvá. A bróker szolgáltatást szintén proxy-n keresztül tudja a kliens használni. A kliens választhat közvetlenül a LUS szolgáltatásból egy számára alkalmas erőforrást, vagy igénybe veheti a bróker szolgáltatást, amely a feladat-követelmények, illetve az erőforrás-képességek illesztését helyette elvégzi.

3.4. Router szolgáltatás

Nem tartozik a Jini alapszolgáltatásai közé, viszont mivel a Jini alapvetően multicast IP forgalomra alapozza a felderítő szolgáltatását, és mivel a multicast forgalmat nem mindenütt engedélyezik, szükség van a multicast üzeneteken terjedő lookup üzenetek továbbítására unicast hálózaton is. A multicast-unicast konverziót végzi a router szolgáltatás.

4. A gLite szolgáltatásai

A gLite az Enabling Grid for E-sciencE (EGEE) projekt során kialakított middleware, amely ma a világ legnagyobb grid rendszerét az elsősorban nagyenergiájú fizikai kutatások céljára létrehozott EGEE-infrastruktúra elemeit köti össze. Maga a gLite az Large Hadron Collider Grid (LCG) köztesrétegből fejlődött ki, az abban szerzett tapasztalatok újragondolásával, magának az LCG köztesrétegnek az újraírásával.

A gLite szolgáltatások halmazát definiálja, amelyek közül nem mindegyiket használjuk ki. Attól függően, hogy egy adott erőforrásra milyen szolgáltatások kerülnek, beszélhetünk EGEE/LCG komponensről, mint például számítási erőforrás (CE), adattárolási erőforrás (SE), stb. A gLite szolgáltatásait, rendszer szinten, az alábbi csoportokba osztja:

- hozzáférési szolgáltatások (Application Programming Interface, API),
- biztonsági szolgáltatások (Authentication-Authorization-Accounting, AAA),
- információs és monitorozási szolgáltatás (GIIS, BDII, RGMA),
- adattárolási szolgáltatások (MDC, RC, SRM, GridFTP),
- feladat-menedzsment szolgáltatások.

4.1. Autentikációs szolgáltatás

A gLite autentikációs szolgáltatása a Globus Toolkit rendszereknél megszokott X509 alapú autentikációs menetet követi, illetve azt igyekszik kiegészíteni az alábbi funkciókkal:

- visszavonási listák vizsgálata, amely arra szolgál, hogy a szolgáltatások minden időpillanatban meg tudjanak győződni egy on-line lekérdezéssel arról, hogy létezik-e az adott entitás, illetve az általa felmutatott tanúsítvány érvényes-e;
- kártya-alapú tanúsítvány-tárolás, amelynek lényege az, hogy a felhasználók a tanúsítványaikat, illetve a privát kulcsaikat fizikailag védett, vagy jól védhető helyen tárolják.

4.2. Authorizációs szolgáltatás

A jelenlegi gLite/LCG rendszerekben az authorizáció elsősorban hozzáférési listákon alapul. Minden erőforrás rendelkezik egy listával, hogy mely felhasználók számára engedélyezi az adott erőforrás használatát. A felhasználókat a tanúsítványukban szereplő megkülönböztető név (Distinguished Name, DN) alapján azonosítják, és a DN – lokális felhasználó párosokat egy adott virtuális szervezeti egységen belül (Virtual Organization, VO) az erőforrások valamilyen protokollon keresztül (web-letöltés, FTP) egymással kicserélik, pontosabban letöltik egy közös forrásról, ami gyakran egyedi.

A gLite majdani authorizációs szolgáltatása az ágens-modellre épül, ami röviden összefoglalva azt jelenti, hogy a felhasználó – erőforrás – hozzáférési-stratégia hármast

egy külső szervezet, a jogosultság-hatóság (Authorization Authority, AA) gyűjti és ellenőrzi. Az AA elsősorban külső összerendeléseket tárol, azaz a felhasználókat csoportokba osztja, és csoport jogosultságokat rendel hozzá a látókörébe eső erőforrásokhoz. A helyi erőforrás egy speciális szolgáltatás, a hozzáférési-stratégia kombináló szolgáltatás segítségével dönti el, hogy a több forrásból érkező, egyazon felhasználóra, erőforrásra érkező hozzáférési szabályok alapján a felhasználót beengedje-e az erőforrásra vagy nem.

Az authorizációs szolgáltatásból jelenleg a Virtual Organization Management System (VOMS) működik [15], amely a felhasználói identitáshoz tud a proxy tanúsítványon keresztül járulékos jogosultságokat rendelni. A VOMS egy relációs adatbázis, amely a felhasználókat és a csoportokat párosítja. Ezt az LDAP adatbázist az erőforrások naponta lekérdezik, és ez alapján készítik el a helyi hozzáférési jogosultságokat szabályozó bejegyzéseket (grid-mapfile). A VOMS adatbázisa, kliens kérésre, a felhasználó tanúsítvány bemutatása után visszaadja az adott felhasználóhoz tartozó szerepköröket, csoport-attribútumokat, amelyeket a felhasználó proxy tanúsítványa tartalmaz majd.

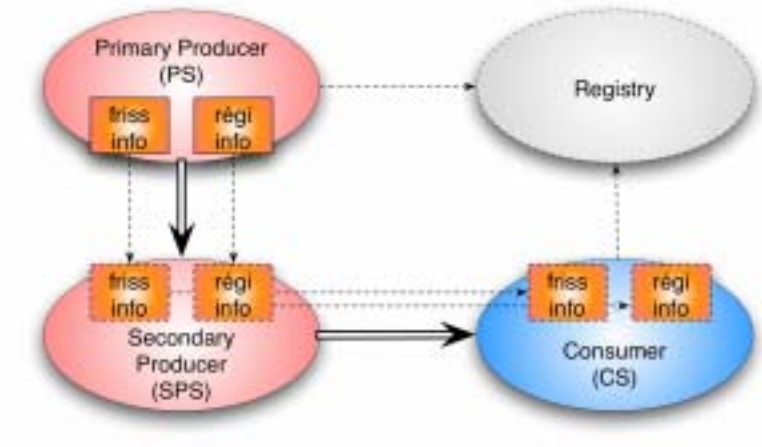
4.3. GAS szolgáltatás

A Grid Access Service (GAS) szolgáltatás legfontosabb feladata, hogy egy belépési pontot adjon a felhasználónak a grid rendszerbe. Amikor a felhasználó hozzá szeretne férni a grid szolgáltatásaihoz, akkor létrehoz egy proxy tanúsítványt, amellyel együtt létrejön egy távoli GAS szolgáltatás. A GAS legfőbb feladata, hogy a felhasználói API által küldött kéréseket fogadja, valamint továbbítsa a megfelelő szolgáltatás felé. A felhasználói GAS szolgáltatás épp addig él ameddig a felhazsnáló proxy tanúsítványa.

4.4. A gLite információs rendszere

A gLite alapvetően relációs adatbázison alapuló információs rendszert használ, amely egy lehetséges implementációja a GGF GMA [23] ajánlásának, és amely három alapvető szolgáltatásból áll (4.1. ábra):

- információ előállító szolgáltatás (Producer Service, PS),
- másodlagos információ szolgáltató (Secondary Producer Service, SPS),
- információ fogyasztó szolgáltatás (Consumer Service, CS).



4.1. ábra: a gLite információs rendszere

A PS szolgáltatás legfontosabb feladata az erőforrás-információ kinyerése, és tárolása. A szolgáltatás a tárolt információt megkülönbözteti az alapján, hogy friss információról, vagy korábbi (történelmi) információról van-e szó, és ezt két egymástól független adatbázisban/táblában tárolja. Friss információ akkor kerül át a "történelmi" információk közé, ha az információ érvényességének maximális időtartama (retention time) lejárt.

A másodlagos információ-szolgáltatók olyan entitások, amelyek a PS források felé fogyasztóként, a fogyasztók felé szolgáltatóként viselkednek. Elsődleges feladatuk a felhasználók geográfiai, illetve adathálózati "közelségében" tárolni a távoli erőforrások által biztosított információkat.

A fogyasztó (CS) alapvetően kétféle eljárással juthat információhoz:

- folyamatos tájékoztatás, azaz a fogyasztó feliratkozik valamely PS, vagy SPS kommunikációs listájára, és az a friss bejegyzésekről folyamatosan tájékoztatja a fogyasztót ("push modell"),
- lekérdezi a szolgáltatót ("pull modell").

A lekérdezés, amely alapvetően SQL protokoll segítségével történik, irányulhat a friss információ lekérdezésére, illetve múltbeli adatok lekérdezésére egyaránt. Az információs rendszer részletes specifikációja, web-szolgáltatás alapú interfészeinek leírása, valamint az adatséma definíciója a [16], illetve a [22] dokumentumban található.

4.5. Számlázási szolgáltatás

A gLite számlázási alrendszere az alábbi főbb modulokból áll:

- Helyi erőforrásmenedzser (LRMS) specifikus adatgyűjtő modulok, amelyek legfőbb feladata az erpőforrás információk begyűjtése.
- Mérés-absztrakciós modul, amely rendszer-független absztrakt leírónyelvre konvertálja a felhasználási információkat tároló rekordokat.
- Gyűjtő modul (Accounting Module), amely elvégzi az absztrakt rekordok összegyűjtését, és lekérdezhetővé teszi azokat.
- Számlázó modul, amely bizonyos virtuális szervezetekre jellemző metrikák alapján kiszámlázza az erőforrás használatot.

4.6. Feladat futtatási szolgáltatás

A feladat futtatása alapvetően a számítási csomópontok feladata, és gyakorlatilag egyetlen szolgáltatása. E szolgáltatás feladata:

- feladatok futtatása, felfüggesztése, újraindítása, leállítása,
- feladatok státuszának nyilvántartása, és lekérdezhetővé tétele,
- feladatok státusza illetve azok változása alapján figyelmeztető üzenetek küldése,
- a helyi erőforrás-menedzserrel (LRMS) történő kapcsolattartás.

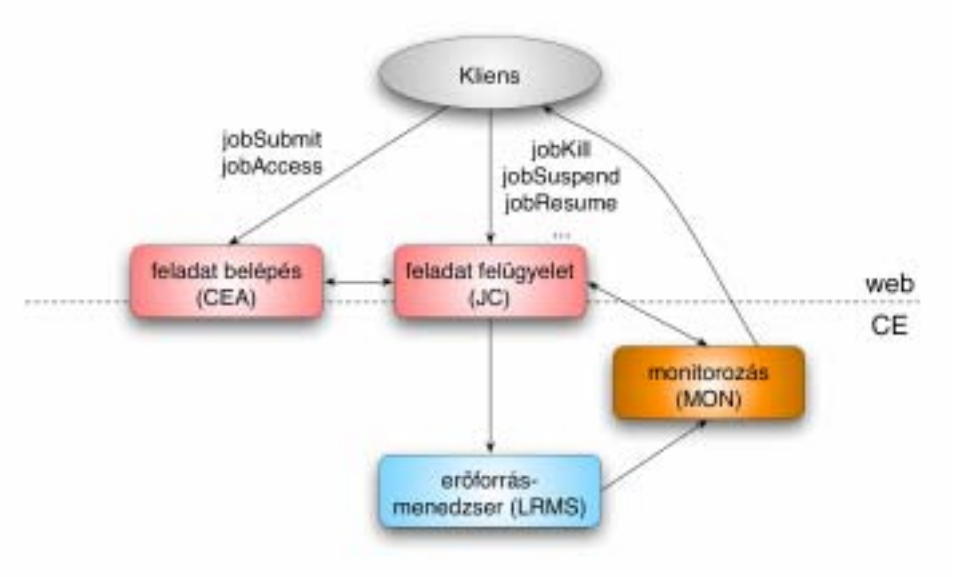
A gLite feladat menedzsment környezete támogatja, mind a "pull", mind a "push" típusú feladatvégrehajtást. Előbbi esetben egy ütemező szolgáltatás keres megfelelő erőforrást, és fordul annak feladat-menedzseréhet (Job Manager, JM) a feladat futtatása céljából, utóbbi esetben maga az erőforrás fordul az ütemező szolgáltatáshoz futtatandó feladatot kérve.

A feladat futtató szolgáltatás alapvetően két szolgáltatásból áll:

- a számítási erőforrásra befogadó szolgáltatás (Compute Element Acceptance, CEA), valamint

- a feladat felügyelő (Job Controller, JC).

E kettő viszonyát, illetve ezek LRMS-hez fűződő kapcsolatát a 4.2. ábra szemlélteti.



4.2. ábra: A feladat futtató szolgáltatás komponensei.

A CEA, amely egy web-szolgáltatás felülettel rendelkező komponens, fogadja a feladatot, és ellenőrzi, hogy a feladat helyben futtatható-e. Ha nem, visszautasítja, ha futtatható, akkor a feladatot leképzi, helyi jogosultságokra, és az így kialakult környezetet átadja a szintén web-szolgáltatás felülettel rendelkező JC modulnak, amely a klasszikus feladat-kezelési feladatokat valósítja meg, azaz fogadja a kliens feladat-felfüggesztés, feladat-leállítás, stb. műveleteit, illetve ezeket leképzi a LRMS felé. A feladat futását egy harmadik komponens, a monitorozó szolgáltatás figyeli, amely az LRMS-ből vett aszinkron üzeneteket továbbítja a kliens felé. "Pull" üzemmódban e szolgáltatás fordul az ütemező felé feladat kérése céllal. Mindhárom szolgáltatás web-szervíz interfésszel rendelkezik, belső kommunikációs protokolljuk nem publikált. A feladat specifikálására a JDL nyelvet használják, amelynek specifikációja a [27] dokumentumban található.

4.7. Feladat ütemező

A feladat ütemező (Workload Management System, WMS) amely a feladat futtató szolgáltatással közel megegyező felülettel rendelkezik. E szolgáltatás képes a futtatásra szánt feladatokat fogadni, és egy feladat-listán ütemezése erejéig tárolni. Rendelkezik az információs rendszerből kinyert, csak olvasható erőforrás-információval, valamint egy feladat monitorozó szolgáltatással. Ezek alapján az információk alapján egy ütemező modul összerendeli a feladatokat az erőforrásokkal. Az ütemező kétféle ütemezési stratégiát ismer:

- a laza ütemezést, illetve
- a mohó ütemezést.

A laza ütemezés azt jelenti, hogy az ütemező inkább vár arra, hogy valamelyik erőforrás feladatot kérjen, míg a mohó ütemezés esetén az ütemező erőforrást keres, és ha talál azonnal átadja a feladatot a megfelelő feladat-futtató szolgáltatásnak.

A moduláris felépítésű feladat ütemező, amely képes többféle ütemezési stratégiát megvalósítani, szintén JDL nyelven leírt feladatokat képes feldolgozni. A WMS szolgáltatás leírása a [21] dokumentumban található.

4.8. Csomagkezelő szolgáltatás

A csomagkezelő szolgáltatás (Package Manager, PM) egy, a számítási csomópontok feladat-felügyeleti szolgáltatását kiegészítő web szolgáltatás. Legfontosabb feladata, hogy egy absztrakt csomagkezelő leírást ad, a ma népszerűen elterjedt csomagkezelőkhöz: RPM, DPKG, PKG_, stb. E web szolgáltatáson keresztül, megfelelő jogosultságok birtokában az erőforrásokra csomagok telepíthetők, onnan csomagok törölhetők, illetve ott frissíthetők.

A szolgáltatás alapvetően arra a koncepcióra épül, hogy általában igaz az, hogy egy VO felhasználói hasonló szoftverkörnyezetet kívánó feladatokat futtatnak, így egy VO egy PM tárházat üzemeltet, ahol mind forrás, mind több architektúrára lefordított bináris csomagok találhatók. Ha egy feladat futásához valamilyen csomagra van szükség, akkor ezt az információt a JDL alapján a JC feldolgozza, és a feladat indítása előtt, a PM kliens oldala, szintén megfelelő jogosítványok birtokában gondoskodik arról, hogy az adott csomag a PM szerverről letöltődjön, és az erőforráson települjön.

4.9. Adattárolási szolgáltatás

A gLite az adattárolási csomópontokat alapvetően kétféle csoportra osztja:

- taktikai adattárolók, illetve
- stratégiai adattárolók.

A taktikai adattárolók elsősorban adatok ideiglenes tárolására szolgálnak. Közös jellemzőjük, hogy általában költséghatékonyan telepíthetők, és a számítási erőforrások hálózati közelségében találhatók. Azokra az adatokra, amelyeket itt tárolunk, semmilyen garanciát nem kapunk, hogy azok biztonságosan meg is maradnak.

A stratégiai adattárolók ezzel szemben azok a tároló-erőforrások, amelyek viszonylag biztos környezetet nyújtanak, magasabb fokú rendundanciával rendelkeznek, és adatok permanens tárolására szolgálnak. Természetesen ez azt is jelenti, hogy kevésbé költséghatékonyak, mint a taktikai tárolók.

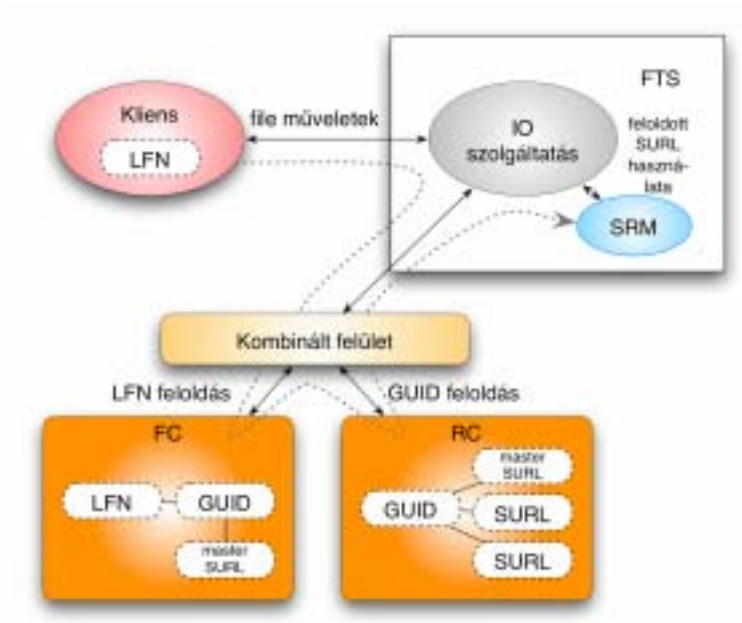
E felosztás abból a megfontolásból ered, hogy mivel a kétféle megoldást egymástól eltérő célokra használjuk, azok felülete is különbözik. Taktikai tárolókat általában futó feladatok használnak az eredményeik átmeneti tárolására, míg a stratégiai csomópontokat inkább tároló-területet menedzselő szolgáltatások az eredmények végleges eltárolására. Ettől függően előbbinek elsősorban a folyamatos adatfolyam fogadását, illetve visszaadását, utóbbinak a már lezárt állományok mozgatását, illetve menedzselését kell támogatnia. E koncepcióra illeszkedik a kétféle megoldás interfésze:

- Storage Resource Manager (SRM) a stratégiai adattárolók esetén, illetve a
- POSIX-interfész taktikai tárolóknál.

Az SRM interfész részletes leírása a [31] dokumentumban található.

A gLite stotage menedzsment szolgáltatása a UNIX file rendszerhez hasonló struktúrát épít ki, de úgy, hogy annak egyes komponensei fizikailag másutt találhatók, és amelyeket az adattárolók adathálózati kommunikáció útján érnek el. Egy állományt a felhasználó általában logikai állománynéven lát (Logical File Name, LFN), amely a file UNIX file rendszerben megszokott abszolút elérési paramétere. A felhasználó a kliensen

a viszonylag könnyen megjegyezhető LFN segítségével azonosítja állományát. (Arról megoszlanak a vélemények, hogy érdemes-e a logikai állományneveket egy egységes file rendszerben láttatni.) Minden LFN-hez tartozik egy globálisan egyedi azonosító (Globally Unique IDentifier, GUID), amely a UUID mechanizmuson alapul. A LFN, illetve GUID közötti megfeleltetést az állománykatalógus (File Catalog, FC) végzi. Az FC azontúl, hogy összerendeli a logikai neveket, illeve a globális azonosítókat, minden egyes bejegyzésről tárol egy fizikai elérést is (Site URL, SURL), amely az állomány tényleges fizikai elérhetősége. Ez az ún. master SURL. A redundancia növelése érdekében minden egyes állománynak lehet több előforduálása, replikája. Nagyon fontos, hogy a replikát megkülönböztessük a másolattól: egy állomány másolata külön állomány, amelynek tartalma megegyezik az eredetivel, két külön kezelhető entitás. Egy állomány replikája azonban a rendszer által biztosított redundancia-szolgáltatás: ugyanaz a file, aminek egyes előfordulásai, replikái explicit módon kezelhetők csak külön. Az előfordulások között létezik egy master SURL, amelynek speciális jogosítványai vannak, például csak őt lehet írni, módosítani, stb. Az állományok GUID, illetve több SURL paraméterét a replikakatalógus (Replica Catalog, RC) rendeli össze. E két szolgáltatást gyakran egy felületen, a kombinált katalógus szolgáltatáson lehet igénybe venni (4.3. ábra).



4.3. ábra: gLite storage szolgáltatás komponensei: az IO kapcsolat, valamint a katalógus szolgáltatások.

Az adatok átvitelét az adatátviteli szolgáltatás (File Transfer Service, FTS) végzi. A felhasználó elsősorban ezzel a web-szolgáltatással áll kapcsolatban. A FTS alapvetően egy várakozási sor, amelyben a SE rögzíti a hozzá beérkezett adatátviteli kéréseket. Az FTS egyrészt a felhasználótól, másrészt egy másik szolgáltatástól az állomány-elhelyező szolgáltatástól (File Placement Service, FPS) kaphat kéréseket. Ez utóbbi, egy virtuális szervezet szintű központi szolgáltatás az adatütemező (Data Scheduler, DS) periodikus lekérdezésével észleli, hogy a gridben valahol valaki nem kezdeményezett-e adatátviteli kapcsolatot az adott csomópontra. A FPS szolgáltatás felelős a visszafelé irányuló kommunikációért is: azaz ha valaki az adott SE csomópontról szeretne adatátviteli kapcsolatot kezdeményezni egy távoli SE csomópontra, akkor egy meghatározott API felületen keresztül (File Transfer Library, FTL) kezdeményezi az adatátvitelt, amit, sikeres autentikáció, authorizáció után a FPS bejegyez a DS szolgáltatásba.

Mivel egy SE több virtuális szervezetet is ki tud szolgálni a beérkező kéréseket egységes csatornába kell transzformálni, illetve a kifelé menő kéréseket demultiplexálni kell. Ezt támogatja a Transfer Fetcher (TF) szolgáltatás. A TF, FPS, illetve a katalógus szolgáltatások site-szintű szolgáltatások, a DS virtuális szervezet szintű, míg az FTS, illetve az SRM csomópont szintű szolgáltatások.

Látható tehát, hogy a gLite storage kezelése rendkívül átgondolt megoldás, ráadásul az adatátvitelre szabványos felületeket, valamint web-szolgáltatás interfésszel ellátott elemi komponenseket használ.

5. Az ARC szolgáltatásai

Az Advanced Resource Connector (ARC) grid köztesréteget a NorduGrid Együttműködés (NorduGrid Collaboration) fejlesztette ki és használja, éles üzemben, 2001-óta [4]. Az ARC jelenleg több, mint 6000 számítási csomópontot kapcsol össze világszerte egyetlen nagy grid rendszerré.

Az ARC szoftver alapvetően Globus Toolkit 2 verziójú rendszer könyvtáraira épít, de az ott kialakított szolgáltatások minőségileg feljavított változatait használja. Nincs például GRAM, Globus-jobmanager, Globus MDS séma, illetve Globus GridFTP. Használja viszont a GT 2 GSI modelljét.

Az ARC köztesréreg szolgáltatásai:

- Grid-manager,
- ARC információs és indexelő szolgáltatás (GIIS),
- Felhasználói interfész és bróker,
- ARC GridFTP,
- Storage-kezelő felület (Smart Storage Element, SSE),
- Monitorozó és loggoló szolgáltatás.

5.1. Grid manager szolgáltatás (GM)

A Grid Manager (GM) szolgáltatás legfontosabb feladata, hogy egy felhasználó által a grid rendszerbe feladott feladatot (job) egy erőforrás fogadni tudjon, illetve továbbítani tudja azt a helyi erőforrás menedzser (Local Resource Management, System, LRMS) felé. Szintén a szolgáltatás feladata, hogy egy épp futó feladat végrehajtását kontrollálja az LRMS rendszerrel történő kapcsolattartás segítségével.

Az ARC GM támogatja:

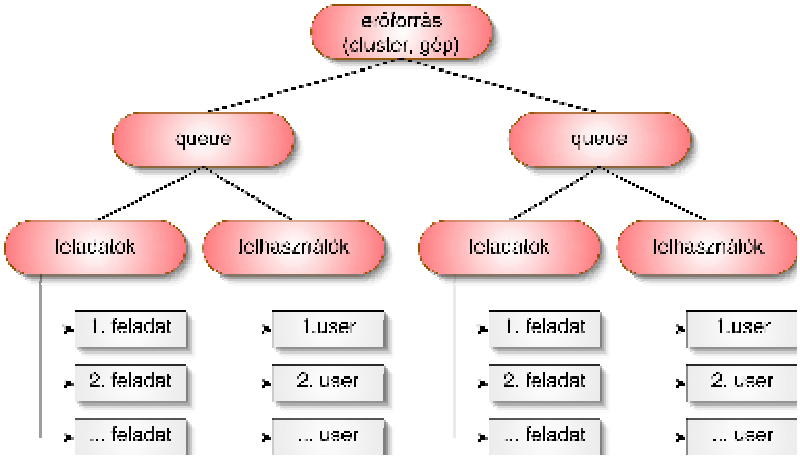
- a "staging" funkcionalitást, azaz a feladat, illetve a hozzá tartozó adatállományok fizikai mozgatását, másolását,
- a replikakezelést (RC),
- a cache-elt állományok megosztását,
- a felhasználó tanúsítványában található DN alapján a felhazsnálók egyszerű authorizációját,
- helyi file rendszer hozzáférést,
- a Globus RSL feladatleírást,
- az egyes feladatok szeparált futtatási környezetben való futtatását.

Fontos különbség a Globus rendszer GRAM megoldásától, hogy a GM nem egy saját belső protokollt, hanem a GridFTP [3] felületet használja a feladatok feladására.

5.2. ARC információs és indexelő szolgáltatás (GIIS)

Az ARC információs rendszere alapvetően a GT 2 szoftver LDAP-alapú GRIS, illetve GIIS komponensei, de attól teljesen eltérő LDAP sémát használ, és máshogy építi fel az információs indexét, mint a Globus. Az ARC LDAP sémája két filozófiai alapgondolaton nyugszik:

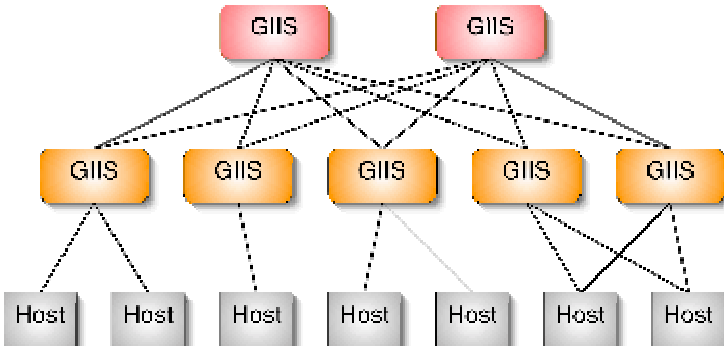
- a lokális adatbázis alapvetően a grid entitásokról, grid felhasználókról, grid feladatokról, és grid erőforrásokról tárol információt (a sémát mutatja az 5.1. ábra),
- a magasabb szintű indexelő szolgáltatások csak referenciát tárolnak az erőforrás-, felhasználó-, queue-, illetve feladat-adatokról.



5.1. ábra: Az ARC helyi erőforás-információs sémája

A GT 2 MDS megoldásához hasonlóan az ARC is rendelkezik információgyűjtő modulokkal, amelyek lehetővé teszik a helyi erőforrás-menedzserből való információ-kinyerést, illetve ennek propagálását a helyi LDAP adatbázisba. A helyi adatbázisok eltárolják a kapott információt, az ARC LDAP séma alapján, amelyeket egyszerű LDAP-lekérdezésekkel el lehet érni. A tárolt adatoknak, ahogy a GT 2 GRIS rendszerében, itt is van egy elévülési ideje, amelyen túl az adatok érvénytelenek.

Az helyi adatbásisok regisztrálhatják magukat egy egységes erőforrás-azonosító (Uniform Resource Identifier, URI) hivatkozással az index szolgáltatásba, amelyek aztán tovább regisztrálhatnak más index szolgáltatásokba. Ily módon egy index-hierarchia építhető fel (5.2. ábra), amelyben, a DNS szolgáltatáshoz hasonlóan léteznek ún. top-level szolgáltatók, országos, vagy regionális szintű szolgáltatók, és intézményi, vagy erőforrás-szintű index szolgáltatók is.



5.2. ábra: ARC Index szerverek hierarchiája

Ha egy felhasználó erőforrást keres az ARC kötésréteget használó rendszerben, akkor előbb a top-level index szolgáltatásokhoz fordul, akik továbbítják a kérését az erőforrás információ szolgáltató felé. Általában igaz, hogy a rendszerben tárolt információ struktúrája illeszkedik a valós geográfiai felépítéshez: egy országon, vagy régión belül található erőforrások az adott ország vagy régió index szolgáltatásába regisztrálják magukat, akik aztán tovább regisztrálnak a top-level szolgáltatások felé. A rendszer robusztusságát az biztosítja, hogy több top-level, illetve országonként/réiónként több index szolgáltatás is működhet.

Az ARC információs rendszere saját LDAP sémát használ, amelynek leírása a [5] dokumentum C függelékében található; a rendszer LDAP protokol használatával kérdezhető le. Az LDAP séma saját OID tartományt használ. Az ARC információs rendszere OpenLDAP alapú, amely tartalmazza a Globus MDS által tartalmazott legfontosabb változtatásokat is [7].

5.3. Felhasználói interfész és bróker szolgáltatás

Az ARC bróker szolgáltatása egy teljesen decentralizált szolgáltatás, amely minden felhasználói felületen megvan: minden olyan gép, amely a felhasználói felületet futtatja egyben egy egyszerű alapelven működő erőforrás bróker is.

A jelenleg parancssoros felhasználói felület a feladatok futtatásához, és az eredmények kinyeréséhez szükséges lefontosabb parancsokat tartalmazza. Az alaputasítások kiegészülnek járulékos utasításokkal, mint amelyek például a storage rendszer kezeléséhez is szükségesek.

A bróker az alábbi elven működik: Ha a felhasználó egy feladatot felad a rendszerbe, akkor a feladathoz mellékel egy feladat leírást, amely a feladatra, illetve a futási környezetére tartalmaz előírásokat (CPU száma, architektúra, input állományok helye, stb.). A bróker előbb keres egy megfelelő erőforrást (GIIS-ből), majd az erőforráson ellenőrzi, hogy a feladandó feladat követelményei megfelelnek-e a helyi erőforrás paramétereinek (GRIS információ). Ha igen, akkor a feladatot föladja az erőforráson, ha nem, akkor másik erőforrást keres.

5.4. Storage-kezelő felület szolgáltatás (SSE, GridFTP)

Az ARC tárterület-kezelő mechanizmusa alapvetően három komponensből áll:

- GridFTP szolgáltatás,
- Globus replika katalógus (Replica Catalog, RC), illetve replika meghatározó szolgáltatás (Replica Location Catalog, RLC),
- Smart Stotage Element (SSE).

A GridFTP hasonló a GT 2 verzióban használatos GridFTP megoldáshoz, azzal a fontos különbséggel, hogy támogatja a fontosabb, jelenleg a különböző grid rendszerekben használt authorizációt (VOCE, GACL) [4]. A megoldás az ftp, illetve a gsiftp kommunikációs protokollokat egyaránt támogatja.

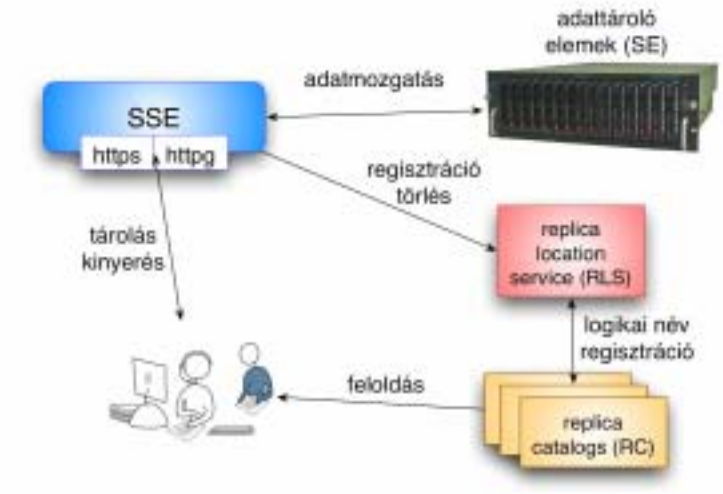
A Globus RC szolgáltatás alapvetően logikai állományneveket rendel hozzá fizikai elhelyezésekhez, logikai állománynév - egyedi azonosító - URL hármas rekordokat tárol, lokálisan. A Globus RLC egy olyan szolgáltatás, amely összefogja a lokális RC-ket, és számukra egy index szolgáltatást tart fenn, amelybe logikai állománynevek, illetve lokális RC bejegyzések kerülnek.

A SSE egy olyan szolgáltatás, amely az előbbi két szolgáltatást gyúrja össze, és egy web szolgáltatás alapú interfészen, HTTPS/HTTPG protokollok segítségével, gSOAP-on keresztül teszi azokat elérhetővé. A SSE saját authorizációs mechanizmussal is kiegészül. Az ARC storage felépítését mutatja az 5.3. ábra.

5.5. Monitorozó és logoló szolgáltatás

Az ARC monitorozó szolgáltatása nem más, mint a GIIS, illetve a GRIS kiterjesztése grafikus web felülettel. Maga a monitorozás is így elsősorban az erőforrások, feladatok státuszának monitorozására (de nem az alkalmazás futási menetének monitorozására) használható. A grafikus felület közvetlenül az információs rendszerből kérdezi le az adatokat, nem tartalmaz cache-elt információt.

A logoló szolgáltatás egy feladat-eredet adatbázisra épül. Ezt az adatbázit a GM szolgáltatás tölti ki a feladott, és futtatott feladat főbb paramétereivel, illetve futási jellemzőivel. Mind a felhasználó, mind a GM meghatározhatja azt, hogy melyik adatbázisba logolódjon a feladat-bejegyzés.



5.3. ábra: Az ARC SSE, komponensei, és a komponensek együttműködése.

6. Szolgáltatás és Web szolgáltatás alapú megoldások

6.1. Szolgáltatások a grid rendszerekben

Bár a jelenlegi grid rendszerek leginkább a számítási erőforrásokra fókuszálnak, ami jól megegyezik az Ineternet által összekapcsolt gépek képével, a grid rendszerek

magában kellene foglalniuk tágabb értelemben vett erőforrásokat is mint az adattárolási, hálózati, szoftver és általános adathalmaz erőforrásokat és számos nem-tipikus erőforrást is mint a grafikus és audio i/o egységeket, manimulátrokat, szenzorokat stb. Ennek ellenére az összes ilyen erőforrásnak a megosztása az Internet segítségével feltételezi hogy az erőforrást valahogy reprezentáljuk egy gépen futó folyamattal (ez alól az egyetlen kivétel lehet a hálózat maga ami így egy problémás grid erőforrás is). Ezek az erőforrásokat reprezentáló folyamatok újabban hálózaton keresztül elérhető grid szolgáltatások.

A szolgáltatás definícója az OGSA [31] szerint: A szolgáltatás egy hálózat eléréssel ellátott entitás ami képességeket tesz elérhetővé. Az OGSA leírásban szolgáltatásokra fókuszálunk: számitási erőforrásokat, tárolási erőforrásokat, hálóztatot, programokat, adatbázisokat és hasonlókat mind szolgáltatásokkal reprezentálunk.

A szolgáltatás orientált megközelítés szerint - ami mára általánosnak mondható - a grid rendszerre tekinthetünk mint kommunikáló szolgáltatások halmazára.

A szolgáltatás orientált megközelítés előnye, hogy a szolgáltatásokat azok interfésze és az általuk használt adatmodell definiálja ami teljesen független azok konkrét implementációjától. Az alkalmazásoknak csak azzal kell törődniük, hogy a szolgáltatások által meghatározott interfészeket és a kommunikációhoz szükséges hálózati protokolokat beszélni tudják.

Az interfész és annak implementációja a szolgáltatások készítői számára is előnyös mivel módjukban áll a teljes szolgáltatás implementációt lecserélni. Mindaddig amíg az új implementáció ugyanazt az interfészt használja, mint korábban aggodalomra nincs okuk.

Az interfész és az implementációs szétválasztása természetesen nem új keletű ötlet. Meglehetősen régóta alkalmazzák az elosztott objektum rendszerben (OPENSTEP/GNUStep DO, Java RMI, CORBA). Az elosztott objetum rendszerek ugyanakkor ezt az elvet az objetumok szintjén alkalmazzák ami jól működhet teljesen új alapokon létrehoztt új alkalmazások esetén de túl alacson szintű megoldás már létező szoftver komponensek összekapcsolása esetén. Az elosztott objektum rendszerek számos objektum orientált tulajdonságot mint például az interfész és az implementáció öröklődéset is támogatják. Ráadásul mivel az interfész definiálása gyakran objetum orientált programozási nyelven történik a legtöbb elosztott objetum rendszer nyelv specifikus.

Magasabb szinten működő paradigma ami ugyanezeket az alapelveket követi a komponens orientált programozás. A komponens orientált progarmozásban az interfész szeparációja és az implementációnak az interfészhez történő kései kötése a két alapelv és elhanyagolja a objetum orientált nyelvek öröklési tulajdonságát. Az interfészek csak egy magasabb szinten definiáltak, amelyeket a komponensek tetszőleges nyelven implementálhatnak, nem feltétlenül objetum orientált módon.

6.2. Web szolgáltatások

Így jól láthatóan a grid manapság elosztott komponens alapú rendszernek is tekinthető. Következésképp a grid alapinfrastuktúrájának támogatnia kell a komponensek közötti kommunikációt, az interfészek defininálásának képességével együtt. Továbbá ennek az alacsony szintnek lehetővé kell tennie, hogy egy kiválaszott interfészhez megfelelő implementációt találjunk. Ezen az alapréteken olyan szolgáltatások építhetőek amik megvalósítják az erőforrások és felhasznlók abasztrakcióját.

Mióta a web techbológia vitathatatlan teret nyert az Interneten elérhető szolgáltatások kimunkálásában, egyre nőtt a web alapú technológiák népszerűsége is.

Sajnos a web-nek vannak árnyoldalai is: például a HTTP-hez kapcsolódó protokollok szinkron állapotfüttelen jellege miatti problémák.

Az utóbbi években számos új web alapú szabvány látott napvilágot ami igyekszik áthidalni a rendszerfejlesztők igényei és a reális lehetőségek közötti szakadékot. Az eredeti egyszerű webtechnológiát mára számos a tradicionális elosztott objetum rendszerekből jövő technikával bővítették ki. Ilyen például az egyre népszerűbb RPC mechanizmus SOAP segítségével. Ezek az új szabványok (WSDL, SOAP, XML stb.) egy új divatszóval: "web szolgáltatások".

A grid technológia nem tud a web alapú technikáktól függetlenül létezni az ipari szereplők nyomása miatt. A korai grid keretrendszerek, mint például a Globus, web szolgáltatás alapú keretrendszerekké formálódtak. Általában tényleg csak keretrendszert készítenek konkrét grid szolgáltatások nélkül így ez az átalakulás nagy áttörést nem hozott a grid technológiában. A grid és a web technológia megtette az első lépéseket egymás irányában de az alapvető kérdések mint például a jogosoltság kezelés, erőforrás leírás stb. továbbra is megoldatlanok.

6.3. Egy web szolgáltatás alapú megoldás: Grid Underground

Grid Underground (rövden GUG) egy szoftver infrastuktúra ami azt a célt tűzte ki maga elé, hogy egyszerű, általános és könnyen kiegészíthető keretrendszert hozzon létre grid szolgáltatások készítéséhez. Ezzel együtt egy minimális szolgáltatás halmazt nyújt amikkel tematikus és regionális grid rendszerek építhetők. A tervezés alapvető irányelveli az egyszerűség, minimalizmus és általánosság volt szemben az OGSA szerű rendszerkben megjelenő funkcióhalmozással és komplexitás növekedéssel.

E tervezési alapelvek mind a grid felhasználók mind az alkalmazás és szolgáltatás készítők, mind pedig az üzemeltetők számára biztosítják a kellő hatékonyságot és egyszerűséget. Az általánosság jegyében természetesen a GUG képes az OGSA szabványoknak megfelelni, de ez a követelmény háttérbe szorul az egyszerű de még működőképes szolgáltatások implementációjával szemben. Ez nem jelenti azt, hogy a rendszer teljesen elhanyagolná a szabványokat például a jelenlegi implementáció is használ számos GGF ajánlást mint a JSDL és a OGSA-BES interfészeket.

A grid széleskörű elterjedésének feltétele, hogy hatékony könnyen telepíthető és kezelhető alapvető grid funcionalitással rendelkező rendszer jöjjön létre, amely könnyen továbbfejleszthető.

A GUG rendszer architektúrája két részre bontható: a szolgáltatások futási környezetét biztosító arhitektúrára és maguknak a szolgáltatásoknak az architektúrájára. Ebben a leírasban a szolgáltatások futási környezetére mint alapkomponensre tekintünk. Minden erőforrás ami a grid részévé válik, az alapkomponenseket futtatja. Az alapkomponensek szolgáltatások életciklusának kezelését végzik, valamint egységes kommunikációs interfészt nyújtanak azoknak. A szolgáltatások valósítják meg az erőforrások képességeit. A szolgáltatások ezeket a képességeket erőforrás leírásokban hirdetik más szolgáltatások felé.

Mivel számos grid szolgáltatást tudunk elképzelni a beépített eszközöktől a szuperszámítógépekig az alapkomponensnek kisméretűnek és, főleg, platform függetlennek kell lennie.

7. A szükséges szolgáltatások, együttműködés

7.1. Szolgáltatások csoportosítása

A különböző grid köztesrétegek által nyújtott szolgáltatásokat alapvetően két kategóriára lehet osztani:

- alapszolgáltatások (core services),
- kiegészítő szolgáltatások (additional services).

Alapszolgáltatás az, amely az adott grid köztesréteg működéséhez feltétlenül szükséges, illetve amelyek nélkül a grid működésképtelen. A főbb alapszolgáltatások:

- autentikációs és authorizációs (AA) rendszer (korábban Grid Security Infrastructure, GSI),
- információs rendszer (Grid Information System, GIS),
- feladat futtató alrenddszer (Grid Resource Allocation Manager, azaz GRAM, Grid Manager, azaz GM),
- feladat ütemező alrendszer (bróker, Super Scheduler)
- monitorozás.

Az alapszolgáltatásokra igaz, hogy általában egymástól eltérő filozófiai alapokon felépített grid rendszerekben is megtalálhatók valamilyen formában.

A kiegészítő szolgáltatások:

- primitív adattárolási szolgáltatás (gridFTP),
- erőforrás bróker,
- monitorozó alrendszer,
- logoló szolgáltatás,
- adattárolás-menedzsment szolgáltatás,
- virtuális szervezetek menedzselésére szolgáló szolgáltatás,
- authorizációs szolgáltatás.

Az együttműködést először az alapkomponensek együttműködése esetén szükséges vizsgálni. Az alapszolgáltatások és a kiegészítő szolgáltatások is megvalósulhatnak több elemibb web-szolgáltatás együttműködésével.

7.2. Együttműködési kezdeményezések

Nagyon érdekes kezdeményezés a Jini és a Globus Toolkit információs rendszerének integrálása [20]. E fejlesztés azt a célt tűzte ki, hogy Jini alól is láthatóvá tegye a GT MDS információs rendszerében tárolt erőforrás információt, illetve a Lookup, Discovery, illetve Registration szolgáltatásokat kiterjessze GT platformra is. Ehhez a CoG [19] keretrendszert használják, amely lehetővé teszi Java platformon az MDS GIIS illetve GRIS alrendszereihez való hozzáférést.

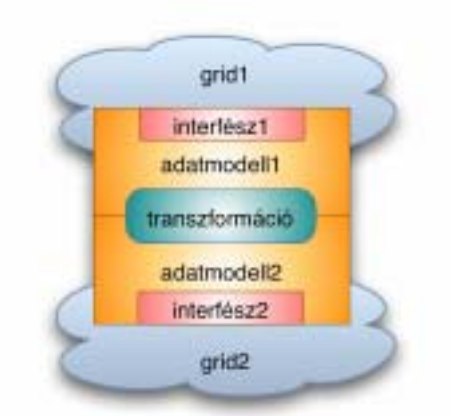
7.3. Az együttműködés elvi sémája, gateway architektúra

Két grid rendszer együttműködésének definíciója: az egyik gridben elérhető alapszolgáltatásokat, transzparens módon elérhetővé tesszük a másik grid rendszerben és viszont. Az együttműködés kiterjeszthető a kiegészítő szolgáltatásokra is.

A megoldás általános sémája az ún. átjáró (gateway) architektúra (GA), amelynek lényege az, hogy a két grid rendszer kapcsolatát egy olyan speciális erőforrás biztosítja, amely a szolgáltatás-kéréseket "átkonvertálja" egyik rendszerből a másik rendszerbe, esetlegesen adat-, illetve kommunikációs-transzformációt is végezve a szolgáltatás-kérésen. A 7.1. ábra illusztrálja a GA sematikus felépítését: mindkét grid minden

szolgáltatása rendelkezik belső adatábrázolással, illetve ezeket az adatokat valamilyen protokollon keresztül kommunikálják klienseik felé: például a Globus MDS esetén ez az adatábrázolás a GLUE vagy régebbi esetben az MDS séma, míg kommunikációs protokollja az LDAP. Ugyanez web szolgáltatás alapú rendszerek esetén XML séma, ahol a kommunikációs protokollja HTML üzenetekbe ágyazott SOAP hívás.

A különböző protokollok segítségével érkező üzeneteket a gateway erőforrás átalakítja egyik megjelenési formából a másikba, úgy, hogy ne történjen információ-vesztés. (Mivel egy "idegen grid" egy speciális erőforrás, így olyan transzformációt is végezhetünk, amely információ-vesztéssel transzformál, ugyanakkor e veszteség a grid-ek viselkedését, a meghozott feladat-erőforrás allokációs döntéseket érdemben nem befolyásolja. Ezt nevezzük aggregációnak.)



7.1. ábra: A gateway architektúra sematikus felépítése.

A gateway erőforrás egy olyan speciális infrastruktúrális elem, amely képes kliensként és szolgáltatóként is viselkedni, attól függően, hogy milyen kérdéssel fordulnak hozzá. Felépítését tekintve többféle grid köztesréteggel is működhet, de legcélszerűbb egy olyan köztesréteget választani, amely kellően moduláris, illetve rugalmas ahhoz, hogy az "idegen protokollokat", illetve más grid köztesrétegekkel való kommunikációt abban könnyen implementálni lehessen.

Egy lehetséges megoldás a Grid Underground (GUG) alapú gateway architektúra, amelynek felépítését a 7.2. ábra mutatja [26].

A GUG megoldás legfőbb előnye együttműködés szempontból az, hogy a belső szerkezete moduláris felépítésű, így a fontosabb alapszolgáltatásaihoz írható olyan modul, amely bár a GUG rendszer belső adatstruktúráihoz natív módon fér hozzá, ezeket az adatokat képes más formátumba transzformálni, illetve kifelé, egy külső protokollon keresztül kommunikálni.

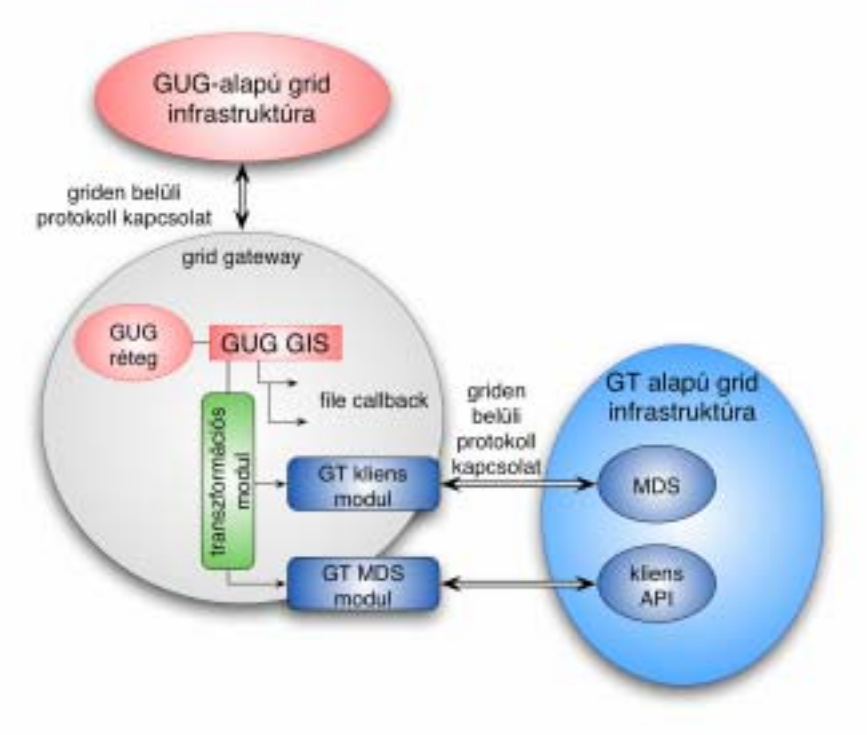
7.3.1. GIS-MDS együttműködés

A 7.2. ábra a GUG és a Globus Toolkit alapú rendszerek információs rendszereinek együttműködésére mutat példát. Az ábrán jól látható, hogy az MDS információcsere két irányú: egyrészt a GT rendszerből el kell érnünk a GUG rendszer információit az ottani kliens API segítségével, másrészt a GUG rendszerből el kell érnünk a GT MDS rendszerét. Ezt a kétirányú kommunikációt két külön GUG modul kezeli le:

1. Az MDS modul a GUG XML reprezentációjú, és SOAP protokollon keresztül kommunikált adatsémáját konvertálja át GLUE sémában tárolt LDAP

reprezentációjú adathalmazra, amelyet LDAP protokollon keresztül kommunikál a GT által használt MDS csatornán.
2. A kliens modul a GT API hívásait használja arra, hogy információt nyerjen ki egy távoli erőforrásból, és ezt az információt, amelyet LDAP reprezentációban kap meg, feldolgozza, és átalakítsa a GUG számára érthető XML reprezentációra.
3. E két modul kiegészülhet egy grid-specifikus transzformációs modullal is.

Nagyon fontos, hogy az adatreprezentáció, valamint a beszélt protokoll szabványos legyen, hiszen csak így garantálható az, hogy minden grid köztesréteg fejlesztő egységes módon értelmezi e transzformációs feladatot.



7.2. ábra: Gateway architektúra GUG alapokon.

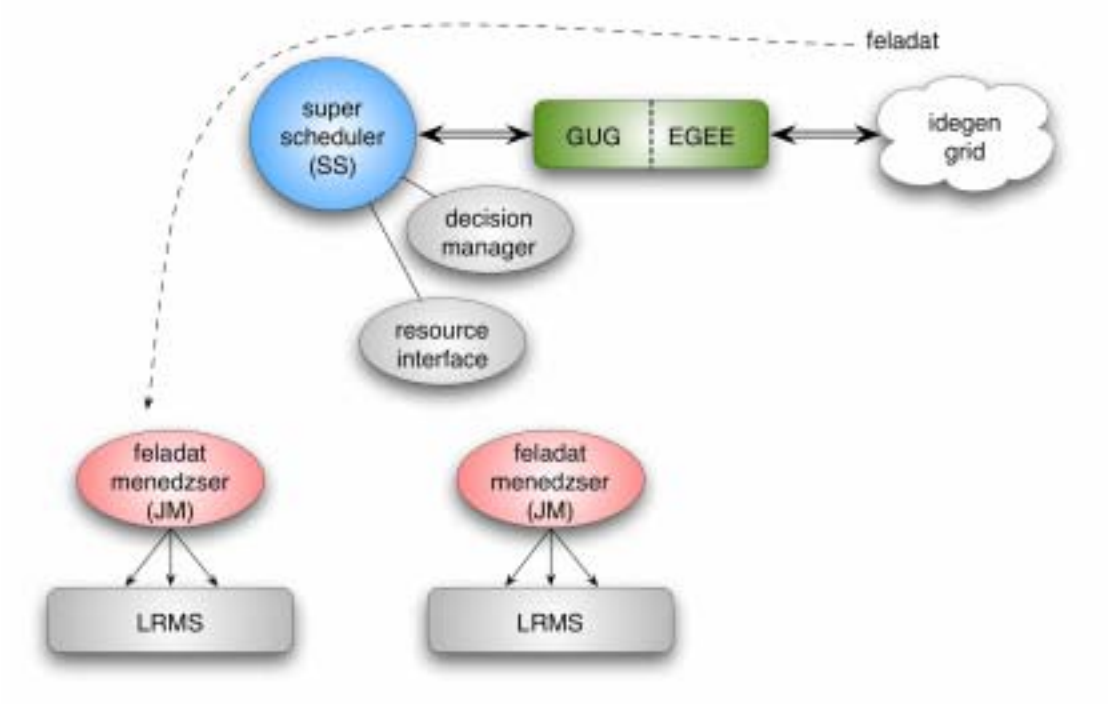
Az információs rendszerek fontos eleme az információ-aggregáció: azaz az átjáró túloldalán található erőforrásokról nem egyedi, hanem aggregált információt közlünk, mintha az egy nagy egységes erőforrás lenne, amennyiben egy bróker ezt az erőforrást választja, akkor az átjáró gondoskodik arról, hogy az aggregált információ alapján hozott döntés "kibontható" legyen a másik grid nem-aggregált információi alapján is. Az aggregálás során, ideális esetben, nem történhet információvesztés.

7.3.2. GRAM-GM együttműködés

A GRAM szintű integrációt szintén két irányra lehet osztani: a GUG felől a GT alapú rendszer felé, és fordítva. Előbbi egyszerűbb, mivel a GUG rendszerben egy erőforrást, illetve az azokon futó feladatokat egy moduláris felépítésű feladat-menedzser szolgáltatás felügyeli. A moduláris felépítésnek köszönhetően külön modulok foglalkoznak a különböző helyi erőforrás menedzserekkel való kapcsolattartással. A külső GT rendszerbe történő feladat-továbbítás is egy ilyen modul segítségével végezhető el, amely nem az erőforrás-menedzsernek, hanem a távoli grid rendszernek adja át a

feladatot. A feladat átadására, a feladatok státuszának lekérdezésére, valamint a feladatok eredményeinek kinyerésére az idegen grid rendszer kliens oldali API-ját használhatjuk.

Idegen rendszer felől, a GUG alapú rendszerbe nehezebb feladatot átadni, mivel a GUG metascheduler architektúrája nem rendelkezik input-oldali modularitással, az OGSA-BES [25] szabványnak megfelelő, JSDL [17] nyelven leírt feladatokat tud fogadni. A kettő közötti illesztést egy speciális szolgáltatás végzi, amely könnyen kialakítható az idegen grid GRAM, GM, stb. alrendszereinek átírásával (7.3. ábra).



7.3. ábra: Feladat átadás GT rendszerből GUG rendszerbe.

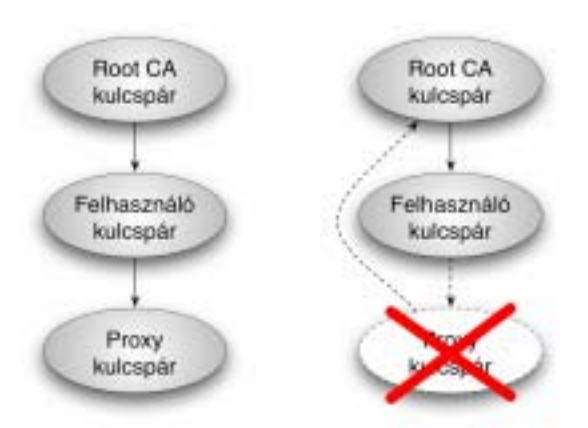
Érdekesség, hogy a GUG ütemezője (SS), valamint a minden erőforráson ott található, az erőforráson futó feladatokat koordináló feladat-menedzser (JM) ugyanazt a feladat-leíró nyelvet érti, mindkettő ugyanúgy képes a feladatokat egyazon felületen elfogadni.

7.3.3. A GSI integráció

A GSI integráció legfontosabb feladata, hogy áthidalja a jelenleg népszerűen használt proxy tanúsítványok, valamint a szintén népszerű felhasználói tanúsítványok autentikációs mechanizmusa közötti különbségeket. A kettő közötti különbséget a 7.4. ábra mutatja.

GT alapú GSI rendszereknél a proxy tanúsítvány egy időlegesen létrehozott entitás, amelyet a felhasználó hoz létre, és ír alá a felhasználói tanúsítványával. Ha egy felhasználó felmutatja a proxy tanúsítványát, akkor a root tanúsítvány hatóságok ezt elfogadják, mint a felhasználó helyében eljáró entitást. GUG esetében nincs proxy tanúsítvány, hanem a felhasználó a kliens interfészen a saját felhasználói tanúsítványát mutatja fel.

Itt az átjárhatóságot egyértelműen a GUG oldalról kell biztosítani, azaz a proxy tanúsítvány-láncoláshoz hasonló megoldást kell biztosítani. Egy ilyen lehetőség a Gridsite [8], amely támogatja a web szolgáltatás felületet, valamint a jogosultság delegációt egyaránt.



7.4. ábra: Felhasználói és proxy tanúsítványok autentikációs lánca.

A GT → GUG irány megoldás a Gridsite rendszer mod_gridsite (VOMS) rendszerén alapul, amelynek az a lényege, hogy képes a felhasználó privát kulcsa nélkül ellenőrizni egy proxy tanúsítvány érvényességét, felülírva ezzel a mod_ssl autentikációját. A fordított irány egyszerűbb mivel csak a proxy tanúsítványokat kell biztosítani: ez is a már korábban említett Gridsite alkalmazással érhető el. A Gridsite alapvetően azt a folyamatot modellezi, amelyet a GSI kliensen végzünk, azaz a létrehozunk egy privát-publikus kulcspárt, amelyet aláírunk a felhasználói tanúsítványunkkal, ezáltal delegálva a identitásunkat a proxy tanúsítvány felé, csak itt annak helye ahol a proxy tanúsítvány létrejön, illetve a felhasználó privát kulcsa tárolódik, különböző fizikai gépen van. A [18] dokumentumban leírt egyszerű delegációs protokollt felhasználva a gateway erőforrás jár el a GUG felhasználó nevében az idegen grid erőforrásainak használata során.

8. Aggregáció

8.1. Aggregációról általában

Nemcsak grid területen fordul elő az a feladat, hogy adott több információ-forrás, illetve több információ-fogadó, akikre igaz, hogy minden fogadó az összes forrás által biztosított információra kíváncsi. Alacsony számú információ-forrás esetén a feladat nagyon egyszerűen kivitelezhető: minden fogadó megkap minden információt, és azt külön dolgozza fel. Ha az informátorok, és a fogadók száma nagy, a probléma nagyságrendekkel nehezebbé válik. Szintén nehezíti a feladatot, ha az rendszerben információ-továbbítók is léteznek. E kihívás azonban nem újkeletű, sok helyen felmerült már ez a probléma:

- peer-2-peer hálózatok információ továbbításánál,
- multicast hálózatokban kommunikációs csomagok továbbításában,
- vagy akár katonai információs rendszerekben, ahol a szemlélők (parancsnokok) minden időpillanatban aktuális, ha nem is teljes, információval szeretnének rendelkezni a csatában résztvevő katonák, eszközök állapotáról, sorsáról.

Hasonló probléma a grid információs rendszereké, ahol rendkívül nagyszámú információsforrás (erőforrás), valamint információ-fogadó (felhasználó, bróker) található. A feladat nem az, hogy az információ-források aktuális státuszinformációval lássák el a fogadókat, de anélkül, hogy a fogadó az információ minden egyes részletére kíváncsi lenne. A fogadó általában szintetizált-összevont, azaz aggregált információ alapján

szeretne képet kapni a rendszer állapotáról, és amennyiben részletesebb információra lenne szüksége, úgy a számára érdekes "erőforrás-mező közeléből" vesz további információt.

E gondolatmenet felveti az információ időbeli és térbeli lokalitásának problémáját:

- Nagyszámú erőforrás esetén nagy mennyiségű információ cserél gazdát, amelynek, tekintve az átviteli csatornák korlátosságára, van egy átviteli idejük.
- Ez azonban azt is jelenti, hogy mire az információ átvitelre kerül, addigra már lehet, hogy az nem aktuális, az alapján nem, vagy csak részlegesen optimális döntéseket lehet meghozni.

Hogy mennyire lehet aktuális az információ, az attól függ, hogy az hányszor cserél gazdát. Ezt nevezzük az információ lokalitásának, azaz egy fogadó ha közelről kapja az információt, akkor az nagyobb valószínűséggel lesz aktuális, mintha távolabbról, több kézen keresztül kapja azt. Egy döntéshozó azonban nem biztos, hogy döntését egy lépésben hozza meg, hanem elképzelhető, hogy több lépésben. Grid rendszerek esetén a konkrét döntés az, hogy a feladatot melyik erőforráson futtassuk. E döntést meg lehet úgy is hozni, hogy begyűjtjük az aktuális információt minden lehetséges erőforrásról, majd ezek közül kiválasztuk azt, amelyik számunkra a legkedvezőbb. A megoldás hátránya, hogy a "távolról" jövő információ alapján kiválasztott erőforrás már nem biztos, hogy szabad, az oda elküldött feladat hibával tér vissza és ez egész eljárást meg kell ismételnünk. (A döntést kiegészíthetjük egy ún. "beszéljük meg" protokollal, amikor az információ-fogadó megtárgyalja az erőforrással azt, hogy mit szeretne futtatni, és milyen feltételekkel.)

8.2. Hierarchikus döntéshozatal, aggregáció

Eredményesebben járhat el az a döntéshozó, aki a döntését több lépésben hozza meg, mégpedig úgy, hogy nem rendelkezik általános információval, hanem csak a közvetlen környezetéről rendelkezik friss információval, és ettől távolodva fokozatosan romló pontossággal szerez friss információt. A pontatlan információ arra alkalmas, hogy a döntéshozó eldöntse azt, hogy milyen irányban kíván tovább puhatolózni, és mely távolabbi rendszerektől szeretne további, aktuális és pontos információt szerezni. A döntéshozatal metódutának egyes állomásait tovább ismételheti, míg a számára legkedvezőbb erőforrást meg nem találja. E döntéshozatali módszert nevezzük hierarchikus döntéshozatalnak, azt az információ-kivonatolást, amely elemi információkból összetett információt hoz létre, aggregálásnak.

A hierarchikus döntéshozatal, valamint az aggregáció két egymást kiegészítő technológia. Egyrészt mindkettő egyfajta faszerkezetet tételez fel, másrészt mindkettő kezeli a távolság fogalmát: minél távolabbi erőforrást szemlélünk annál összevontabb, ugyanakkor aktuális információ alapján szeretnénk a döntést meghozni.

8.3. Aggregáció grid együttműködések esetén

Ott ahol többféle különböző elven felépített grid rendszer együttműködéséről, átjárhatóságáról beszélünk szintén felmerül az aggregáció gondolata. Természetesen itt is megkülönböztethetünk finomabb granularitású aggregációt a durvábbtól. Finomabb granularitású aggregáció lehet például az, amikor egy grid rendszeren belül is alkalmazzuk ezt a módszert, míg durvább granularitású az, amikor csak a különböző rendszerek között. Ez utóbbi mellett egy igen jó érv, hogy a jelenleg megvalósított grid információs rendszerek mindegyike inkább az egysíkú lineáris információ-feldolgozást támogatja,

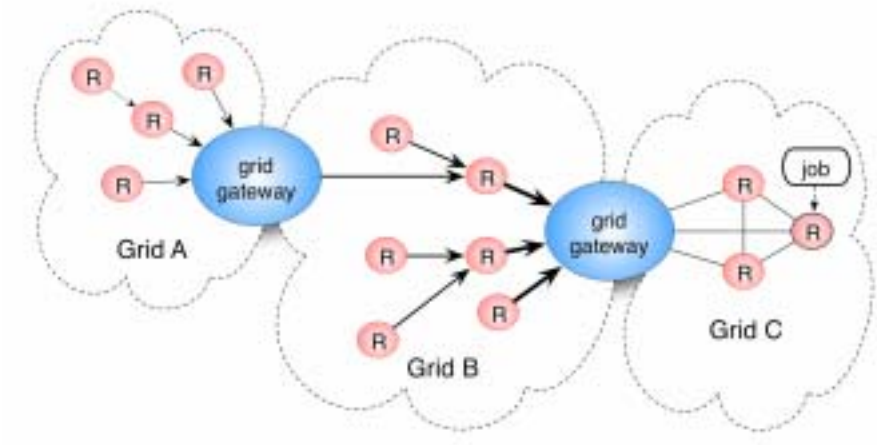
azokba beleerőltetni egy hierarchikus megoldást igen nehézkes lenne. Az is igaz viszont, hogy az idegen grid rendszereket nem szeretnénk teljes részletességükben látni, hiszen egyrészt nagy számú idegen grid rendszerrel állhatunk kapcsolatban, másrészt azok tetszőlegesen komplexek lehetnek.

A 8.1. ábra egy lehetséges információ-terjedési modellt mutat be. Az együttműködő rendszer itt három egymással kapcsolatban álló grid, amelyek közül az A és a B jelű képes arra, hogy az információ cserélők már eleve kivonatolt információt adjanak tovább, a C jelű erőforrásai viszont nem képesek erre, e rendszerben mindenki ugyanazzal az információval rendelkezik. (Ezt szemlélteti a teljesen összekötött gráf.) A C jelű grid rendszerben feladott feladat egyrészt választhatja azt, hogy közeli erőforráson fut le, illetve választhat egy olyan erőforrást, amely ugyan távol van, állapotáról pontatlan információval rendelkezik, ugyanakkor ez az információ friss. Ezt az információt számára a gateway gép gyűjti össze, a B rendszerben információ aggregálással, amely információ jöhet akár az A rendszer erőforrásairól, illetve azok aggregálását elvégző átjáróról. Amennyiben a feladat egy távoli erőforrást használ, át kell haladnia a gateway gépen, amelyen állva további információt gyűjthet be a számára érdekes erőforrásokról, pontos információ kinyerése végett akár közvetlenül le is kérdezheti azokat.

Egy lehetséges módszer az átjáró számára, ha az információ forrásokat egy faszerkezetben látja, akár úgy, hogy az információs összeköttetéseknek a feszítőfáját elkészíti, majd a feladatot, amennyiben az például átjutott a B jelű rendszerbe a fa levelei felé haladva adja át az információ szolgáltató erőforrásoknak. Ezeken az erőforrásokon állva a feladat egyre pontosabb információhoz jut az átjárótól távolabb eső erőforrások állapotáról is. Természetesen azt, hogy a feladat záros határidőn belül találjon magának egy erőforrást rögzíthetjük a rendszerben egy maximális döntési mélység paraméter bevezetésével és fokozatos csökkentésével, mindahányszor a feladat erőforrást vált. (Persze a feladatnak egy adott gridbe belépve nem feltétlenül kell minden erőforrásra fizikailag is eljutnia, az maradhat a gateway gépen.)

8.4. Paraméterek aggregációja, algoritmusok

A grid információs rendszerek által szolgáltatott adatok alapvetően paraméter – reláció – érték hármasok, amelyek szemantikája általánosan elfogadott. A paraméterek elsősorban erőforrás attribútumok, de lehetnek akár absztrakt jellemzők is.



8.1. ábra: Információ-csere három együttműködő grid rendszernél.

A relációk rendszerint sorrendiséget, vagy egyenlőséget reprezentáló relációk. Az értékek sokfélék lehetnek: beszélhetünk numerikus értékekről (CPU szám, memória méret,

kapacitás), string értékekről (erőforrás leírás, architektúra, környezeti attribútumok), illetve logikai értékekről (például egy speciális erőforrás megléte, szolgáltatás megléte, stb.). Nagyon fontos hangsúlyozni, hogy ezeknek az értékeknek külön szemantikájuk van. E szemantikát elsősorban maga a feladott feladat osztálya határozza meg. Azaz másképp értelmezi a "kétszer-tíz processzorból" összevont "20 processzor" erőforrás-információt egy pataméter feltérképezési feladat, mint egy osztott memóriára épülő alkalmazás. Maga a szemantika kifejezhető speciális meta-attribútumok segítségével, amelyek az aggregált információ keletkezésére, illetve jellegére vonatkozhatnak. Például egy speciális "SMP" attribútum igaz, vagy hamis értéke jelezheti, hogy az aggregált CPU információ valójában több egyprocesszoros erőforrást takar, vagy egy nagyobb szuperszámítógépet.

Az grid rendszerekben javasolt információ-aggregációs algoritmusok kétféleképp vonhatnak össze információt:

- mennyiségi aggregáció,
- minőségi aggregáció.

A mennyiségi aggregáció azt jelenti, hogy a több információ-forrásból érkező hasonló jellegű attribútumokból képzünk egy egységes attribútumot. Ilyen összevonás például, ha összeadjuk több erőforrás CPU-inak számát, vagy a rendelkezésre álló háttértároló-kapacitásukat. Mennyiségi aggregációra használhatók a minimalizálás, maximalizálás, összeadás, szorzás, diszjunkció, konjukció, számlálás műveletek, attól függően, hogy milyen jellegű az az attribútum, amelyet össze kívánunk vonni [29]. Ekkor egy erőforrás arról rendelkezik információval, hogy az ő környezete hogyan épül fel, illetve a közvetlen környezetétől távolodva milyen járulékos erőforrások léteznek.

A minőségi aggregáció ezzel szemben erőforrások minősítését jelenti. Azaz az erőforrások különböző jellegű attribútumait összevonva az erőforrás "jóságára" adunk egy számot, kifejezve ezzel azt, hogy az erőforrás a megadott feladat osztály számára mennyire megfelelő. Természetesen egy erőforrás lehet, hogy minden feladat-osztály számára kiválóan megfelel, de lehet olyan is, amely csak egy speciális osztályú feladatok fogadására (például PVM párhuzamos programok számára) lehet alkalmas, más osztályok számára nem. E tématerületet kimerítő elméleti alapokon vizsgálja a [30] jelű szakirodalom, amely több igazoltan optimális algoritmust mutat be az erőforrás minőségi jellemzőinek kivonatolására.

Tekintettel a grid rendszerek jelenleg használatos megoldásaira, elsősorban mennyiségi aggregációt kell használnunk, amely jelenleg a főbb attribútumokra, elsősorban a feladatok leírására vonatkozó szabványokban található attribútumokra kell vonatkozzon.

9. Absztrakt protokollok

A különböző grid rendszerek felépítése szempontjából a különböző kommunikációs csatornákon küldött üzenetek egységes protokollba ágyazására leginkább a web-szolgáltás protokoll-stack használható. E protokoll-stack olyan web fölött futó protokollok gyűjteménye, amelyek segítenek a web szolgáltatások definiálásában, felderítésében. Alapvetően az alábbi protokollok alkotják:

- Szolgáltás Traszport Protokoll, amely az üzenetek biztonságos, web alkalmazások közötti továbbítására szolgál. Ilyenek a HTTP(S), FTP(S), illetve az SMTP(S) protokollok.

- XML Üzenetküldő Protokoll, amely arra szolgál, hogy az üzeneteket XML formába kódolja, és biztosítsa, hogy ezeket a különböző üzenetküldők egyformán értelmezik. Ilyen protokoll az XML-RPC, vagy a SOAP.
- Szolgáltatás Leíró Protokoll, amely arra szolgál, hogy a web szolgáltatás publikus interfészét definiáljuk. Itt leggyakrabban a WSDL nyelvet használják.
- Szolgáltatás Regisztrációs Protokoll, amely segítségével a szolgáltatások magukat, helyüket, főbb jellemzőiket egy egységes regisztrációs adatbázisba vehetik fel. Ez a UDDI.

10. Összefoglalás

E dokumentumban az volt a célunk, hogy áttekintést adjunk a napjainkban használt grid köztesrétegekről, az általuk nyújtott szolgáltatásokról, és a szolgáltatások csoportosításával, tipizálásával meghatározzuk azokat a pontokat, amelyeken keresztük a különféle filozófia mentén kialakított grid rendszerek integrálhatók. Az integráláson itt a grid feladatok, illetve a grid információ átjárhatóságát értjük. Bemutattuk a Globus, Jini, LCG, ARC, illetve általában a web szolgáltatás alapú megoldások esetén nyújtott szolgáltatásokat, és meghatároztuk azokat az alapszolgáltatásokat, amelyeket a különböző rendszereknek mindenképp nyújtania kell, hogy azok működjenek.

Ezek mentén meghatároztuk a lehetséges együttműködés forgatókönyvét, és kialakítottuk a gateway architektúrát (GA), amely a különböző grid köztesrétegek közötti együttműködés egy lehetséges formája. Kialakítottuk az együttműködés rendszertervét, amely bemutatja, hogy a jelenlegi GUG köztesréteghez milyen modulokat kell kifejlesztenünk ahhoz, hogy az képes legyen az "idegen grid köztesrétegekkel" történő információcsere lebonyolítására.

Hivatkozások

[1] http://www.globus.org/toolkit/docs/2.4/mds/Schema.html
[2] http://www.cnaf.infn.it/~sergio/datatag/glue/index.htm
[3] http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf
[4] http://www.nordugrid.org
[5] http://www.nordugrid.org/documents/arc_infosys.pdf
[6] http://www.globus.org
[7] http://www.globus.org/mds/mds2/
[8] http://www.gridsite.org/
[9] http://www-unix.globus.org/api/c-globus-2.4/globus_gram_job_manager/html/globus_job_manager_rsl.html
[10] http://www-unix.globus.org/api/c-globus-2.4/globus_gram_protocol/html/
[11] http://www-unix.globus.org/toolkit/docs/4.0/execution/wsgram/WS_GRAM_Public_Interfaces.html
[12] http://www.sun.com/jini/specs/
[13] http://www.sun.com/software/jini/specs/jini1.2html/discovery-spec.html#2540
[14] https://edms.cern.ch/document/476451/
[15] http://grid-auth.infn.it/docs/VOMS-Santiago.pdf

[16] http://forge.ggf.org/sf/docman/do/downloadDocument/projects.ogsa-wg/docman.root.meeting_materials_and_minutes.ggf_meetings.ggf16/doc13052;jsessionid=3B8C754CA2DE0ADE3AE5AE085409910E
[17] https://forge.gridforum.org/projects/jsdl-wg/
[18] http://www.gridsite.org/wiki/Delegation_protocol
[19] http://www.globus.org/cog/java/
[20] http://tyne.dl.ac.uk/Papers/Rana/rana/node9.html
[21] https://edms.cern.ch/file/490223/3.1/specification.pdf
[22] http://www.r-gma.org/
[23] http://www-didc.lbl.gov/GGF-PERF/GMA-WG/
[24] http://www.rfc-archive.org/getrfc.php?rfc=3820
[25] https://forge.gridforum.org/projects/ogsa-bes-wg/
[26] http://gug.grid.niif.hu
[27] http://www.infn.it/workload-grid/docs/DataGrid-01-TEN-0142-0_2.pdf
[28] http://www.ece.rutgers.edu/~parashar/Classes/04-05/ece572/ds/njiang.pdf
[29] http://ebusiness.mit.edu/research/papers/106%20SMadnick,%20Siegel%20Information%20Aggregation.pdf
[30] http://www.almaden.ibm.com/cs/people/fagin/jcss03.pdf
[31] http://sdm.lbl.gov/srm-wg/