Elosztott adattárolási technikák web szolgáltatás alapú grid rendszerekben

Nagy Zsombor

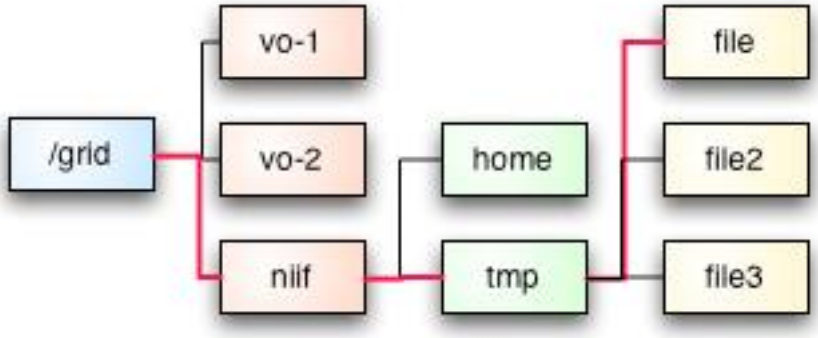
1 Bevezetés

A nagyobb grid rendszerek utóbbi időben történő fejlesztése közben felmerült a határozott igény elosztott tároló rendszerek kifejlesztésére (pl. DataGrid Project[1], EGEE gLite Data Management[2]). Egy elosztott tároló rendszer előnyei közé tartozik a hibatűrés és a nagy rendelkezésre állás, hiszen az adatok tipikusan földrajzilag szeparált tárolókon helyezkednek el több példányban. Másik nagyon fontos szempont egy elosztott tároló rendszer esetében, hogy a különböző helyeken levő adatok megosztására van lehetőség. Másrészt a grid rendszerekben rengeteg kihasználatlan tárterület van, amelyeket egy elosztott tároló rendszer, tárolási erőforrásokká alakíthat, így nem maradnak kihasználatlanul.

Az alábbiakban ismertetendő tároló rendszer implementációja a GUG[6] (Grid Underground) nevű köztesréteg (middleware) része, amely az NIIF Intézet[7] által fejlesztett és üzemeltett új generációs ClusterGrid[8] köztesrétege. A GUG rendszer magja egy web szolgáltatás alapú keretrendszer, mely lehetővé teszi web szolgáltatásos felülettel rendelkező szolgáltatások futtatását, kezeli azok életciklusát, menedzselhetővé teszi azokat. Ebben a keretben fut a tároló rendszer jelenlegi implementációja.

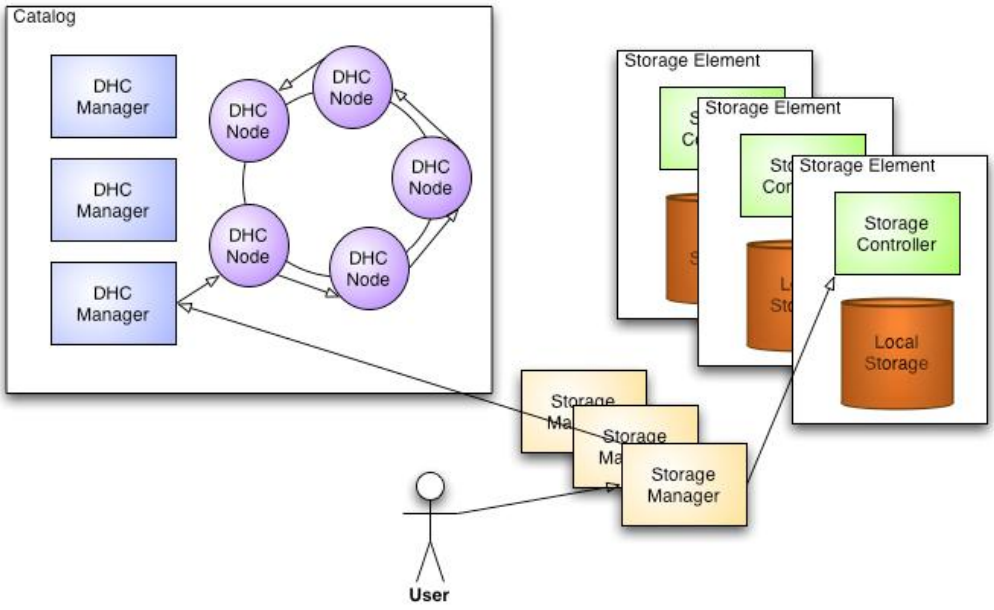
2 A rendszer koncepciója, architektúrája

A tároló rendszer fejlesztésekor fontos szempont volt, hogy a felhasználók számára az elosztottság a lehető legnagyobb mértékben transzparens maradjon. Ezért az egész rendszerben egy egységes névtér, egységes könyvtárstruktúra van, ebben helyezhetik el a felhasználók az állományaikat és könyvtáraikat, amelyekre logikai állománynévvel, LFN-nel hivatkoznak. Egy LFN tulajdonképpen egyéb útvonalaknál megszokott formátumú, per jelekkel ("/") elválasztott nevek sorozata. Egyetlen speciális megkötés, hogy "/grid" névvel kell kezdődnie. Az 1. ábrán látható, hogyan lehet bejárni a logikai fában a "/grid/niif/tmp/állomány" LFN-nel kijelölt útvonalat.



1. ábra: LFN feloldása

A tároló rendszerben a tárolási erőforrásokat Storage Element-nek nevezzük, amely egy leegyszerűsített SRM-szerű[3] felületet nyújt, amelyen keresztül kezdeményezhetjük adatok tárolását és a tárolt adatok visszakérését. A Storage Element egy absztrakt erőforrás, a tényleges megvalósítását egy Storage Controller nevű szolgáltatás végzi. A felhasználók a Storage Manager szolgáltatásokon keresztül tudják elérni a tároló rendszert. A Storage Manager-ek egy elosztott katalógust használnak a könyvtárstruktúra tárolására. Ezt a katalógust kétféle szolgáltatás valósítja meg. A rendszer architektúráját a 2. ábrán láthatjuk.



2. ábra: A tároló rendszer architektúrája

3 Storage Element

A *Storage Element (tárolóelem)* egy absztrakt erőforrás a grid rendszerben. Tárolási kapacitást jelképez, amely képes strukturálatlan adatokat (adott méretű byte-sorozatot) tárolni, és visszakeresni azt. Egy erőforrást akkor nevezhetünk Storage Elementnek, ha megvalósítja a következő funkciókat.

Állomány elhelyezése

Az adatok (továbbiakban állományok) tényleges elhelyezése előtt a *prepareToPut* metódussal kezdeményezni kell azt az állományok méretének meghatározásával. Több állomány feltöltését is lehet egyszerre kezdeményezni, ilyenkor természetesen több állományméretet kell megadni. Ehhez a kéréshez készül egy azonosító token, aminek segítségével később lekérdezhetjük a kérés állapotát. Minden egyes elhelyezni kívánt állományhoz készül egy SURL azonosító, amivel később hivatkozhatunk az adott állományra. És hogy megkezdhessük a feltöltést, minden állományhoz keletkezik egy TURL, ahova ténylegesen feltölthetjük az állományt. Mindezek mellett mindenről kapunk egy állapotjelzést is. Ennek a metódusnak a válasza tehát egy token, egy általános status, valamint minden állományhoz egy saját status, TURL és SURL. Ezek után a kapott tokennel hívva a *statusOfPutRequest* metódust, megkapjuk ugyanilyen formában az aktuális állapotokat.

Állomány visszakeresésese

Ha rendelkezünk egy állomány SURL-jével, akkor kezdeményezhetjük az állomány letöltését a *prepareToGet* metódus meghívásával. Több SURL-t is átadhatunk, amennyiben több állományt szeretnénk letölteni. Ennek a kérésnek a hatására készül egy token, amellyel később statust kérhetünk. Minden SURL-hez készül egy TURL, amelyről ténylegesen letölthető az adott állomány. Visszatérési értékként megkapjuk a tokent, egy általános statust, és minden állományhoz egy saját statust és TURL-t. A kapott tokennel meghívhatjuk a *statusOfGetRequest* metódust, amely ugyanilyen formában adja meg az aktuális állapotokat.

Állomány eltávolítása

Ha rendelkezünk egy állomány SURL-jével, akkor eltávolíthatjuk az állományt a *releaseFiles* metódus meghívásával. Egyszerre több SURL-t is átadhatunk amennyiben több állományt szeretnénk eltávolítani. Válaszként kapunk egy általános status kódot, valamint minden eltávolítandó állományhoz is kapunk egy saját állapotjelzést.

Storage Controller

A Storage Element funkciót a *Storage Controller* szolgáltatás valósítja meg. A rendszerben a Storage Controllert teljes állományok eltárolására használjuk. Feltöltéskor átadva az állomány teljes méretét, majd feltöltve az állományt, kapunk egy azonosítót, amellyel később az állomány letölthető a StC-ről.

Ennek az azonosítónak a neve SURL (Storage URL), tartalmazza a Storage Controller ID-jét és az állomány StC-n belüli azonosítóját, ez így együtt egyedi, egyértelműen azonosítja az állományt (byte-sorozatot) az egész rendszerben.

A byte-ok fel- és letöltése, azaz az adattranszfer nagymértékben független a Storage Controllertől, a jelenlegi implementációban egy webszerveren keresztül történik, de lehetne FTP, GridFTP[4], ByteIO[5], vagy bármi más. Ezt a függetlenséget úgy értük el, hogy bevezettük a TURL (Transfer URL) fogalmát. A TURL egy dinamikusan létrehozott URL, ennek segítségével végezhető el az adattranszfer. Feltöltés esetén átadjuk az StC-nek az állomány méretét, amely erre válaszként nem csak egy SURL-t ad, hanem létrehoz egy TURL-t is, ide kell valójában feltöltenünk az állományt. Letöltéskor átadjuk az StC-nek az SURL-t, amelynek hatására az generál egy TURL-t, amelyről ténylegesen letölthetjük az állományt. A jelenlegi implementációban az adattranszfer HTTP-vel történik GET és PUT metódussal, ezt egy az StC által indított webszerver szolgálja ki. A webszerver URL-jei menet közben keletkeznek, és csak egyetlen letöltést illetve feltöltést szolgálnak ki, aztán megszűnnek létezni.

4 Elosztott katalógus

A tároló rendszernek szüksége van egy kulcs-érték párok tárolására és kulcs alapú visszakeresésére képes hatékony katalógusra. Nem szeretnénk, ha ez egy központosított szolgáltatás lenne, mert az nagyban sebezhetővé tenné a rendszert, de azt se szeretnénk, hogy a katalógus használójának tudnia kelljen arról, hogy a katalógus elosztott-e vagy sem. Ezt a problémát a katalógus manager-ek segítségével oldottuk meg, amelyek egy egységes felületet nyújtanak a katalógus adatait ténylegesen tároló katalógus csomópontokhoz.

A jelenlegi implementációban a katalógust *Distributed Hash Catalog*-nak (DHC) nevezzük, a katalógus csomópontokat *DHC Node*-nak, a katalógus managerek-t pedig *DHC Manager*-nek (DM). A katalógus felhasználói csak a DM-ekről tudnak, és velük kommunikálnak. Azért beszélünk többes számban a katalógus manager-ekről, mert bár valójában csak egyre van szükség a katalógus eléréséhez, de mégis több DM-et használhatunk a rendszerben, amelyek közül bármikor kiválaszthatunk egy tetszőlegeset, így nem okozhat problémát, ha valamelyik DM esetleg nem működik. Bármely DM-en keresztül ugyanazt a katalógust látjuk, és érjük el.

A katalógusnak mindössze két műveletet kell biztosítania, az egyik az új kulcs-érték pár bejegyzése, a másik pedig az egy adott kulcshoz tartozó érték visszakeresése. A jelenlegi implementációban a kulcsok és az értékek is

egyszerű karakterfüzérek. A katalógus egyáltalán nem értelmezi ezeket az adatokat, nem foglalkozik vele, hogy a katalógus felhasználói mit tárolnak bennük. A tároló rendszer valójában XML dokumentumokat tárol a katalógusban értékként.

5 Storage Manager (StM)

A tároló rendszer felhasználója elsődlegesen egy *Storage Manager*-rel találkozik (és csak másodlagosan a Storage Controller-ekkel). A Storage Manager feladata, hogy a felhasználó kéréseinek megfelelően változtassa a katalógust, kezelje a Storage Controller-eket, és az StC által adott TURL-t eljuttassa a felhasználóhoz. StM-ből is lehet több a rendszerben, a felhasználó bármelyiket választhatja, akár kérésenként válthat közöttük.

A felhasználó a tárolón levő állományokat a logikai nevükkel azonosítja (LFN, Logical File Name), ezenkívül a felhasználó könyvtárakat is kezel, amelyeket szintén LFN-nel azonosít. Az StC-k semmit nem tudnak könyvtárakról illetve LFN-ekről, a katalógus egyáltalán nem értelmezi az általa tárolt adatokat, így mindezek kezelése a Storage Manager feladata marad.

Ahhoz, hogy állományrendszert tudjunk építeni, minden egyes állománynak és könyvtárnak kell hogy legyen egy egyedi azonosítója (GUID, Global Unique IDentifier). Ezeket legjobban a hagyományos UNIX állományrendszerek inode fogalmához lehet hasonlítani. A GUID-okat használjuk a katalógusban kulcsként. Tehát ha tudjuk egy állomány vagy könytár GUID-ját, akkor a katalógushoz fordulhatunk, hogy megkapjuk az adatait. Ezeket az adatokat XML dokumentumban tároljuk a katalógusban, ennek az XML dokumentumnak a szerkezete más egy állomány és más egy könyvtár esetén.

Egy állomány esetén tároljuk a létrehozás időpontját, a méretét, azt hogy futtatható-e, és az SURL-eket (lehetséges ugyanis, hogy egy állományt több példányban is tárolunk, hogy a rendelkezésre állást és a hibatűrést megnöveljük).

```
<?xml version="1.0" ?>
<File>
  <Created>1139721259</Created>
  <Size>238</Size>
  <SURL>2fa00259-15b5-42b7-92df-863e5522758f/oduzudu910<SURL>
  <SURL>1bc34212-4323-ab43-54ff-9843d8e723df/ahigimi134<SURL>
  <IsExecutable>1</IsExecutable>
</File>
```

Egy könyvtár esetén is tároljuk a létrehozás időpontját, valamint a könyvtár bejegyzéseit. Egy könyvtárbejegyzés egy nevet és egy GUID-ot tartalmaz, azaz azt mondja meg, hogy az adott könyvtárban létezik egy adott nevű elem az adott GUID-dal.

```
<?xml version="1.0" ?>
```

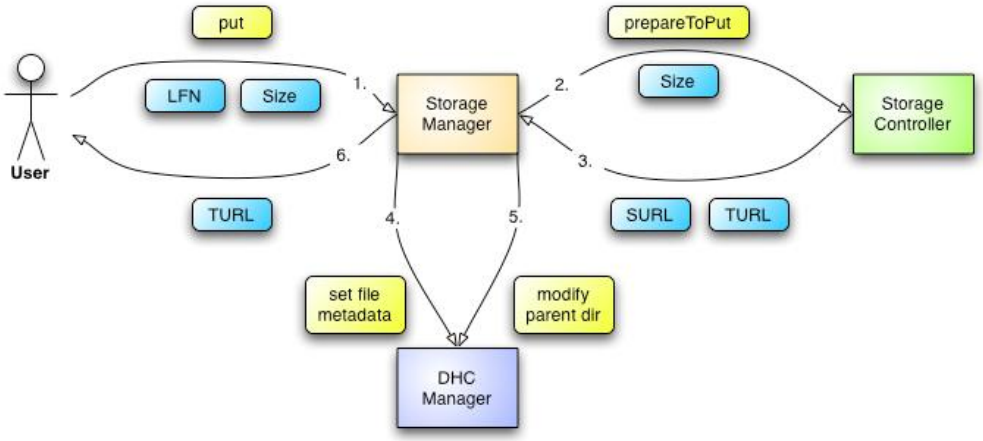
```
<Directory>
  <Created>1139597511</Created>
  <Entry>
    <GUID>lisepal-870</GUID>
    <Name>sheridan.bin</Name>
  </Entry>
  <Entry>
    <GUID>sogimuz-188</GUID>
    <Name>b5.sh</Name>
  </Entry>
</Directory>
```

Látható, hogy az állomány adatainál nem tárolunk nevet, az állomány nevét a könyvtár bejegyzéséből lehet megtudni. A példabeli könyvtárban két bejegyzés található, az egyiknek a neve "sheridan.bin", a másiké "b5.sh". Azt nem tudhatjuk, hogy ezek könyvtárak-e, vagy állományok, ez az információ csak akkor derül ki, ha lekérjük az adott bejegyzés adatait. Pl. ha a "sheridan.bin" nevű entitás adataira vagyunk kíváncsiak, akkor a GUID-jával ("lisepal-870") mint kulccsal kell a katalógushoz fordulni, és a visszakapott XML dokumentum típusa eldönti, hogy állomány vagy könyvtár-e az adott bejegyzés.

Látható ezen kívül, hogy a fenti állományadatokat tartalmazó XML dokumentumban kettő SURL bejegyzés található, ami azt jelenti, hogy az adott állomány két példányban található meg a tároló rendszerben (feltételezhetően két fizikailag, akár földrajzilag is szeparált Storage Controlleren). Ha az egyik SURL által meghatározott StC valamiért nem elérhető, akkor a Storage Manager megpróbálkozik a többi SURL-lel is.

6 Működési példa: állomány feltöltés

Amikor a tároló rendszer felhasználója egy állományt szeretne feltölteni a rendszerbe, akkor választ egy Storage Manager-t, és küld neki egy *put* üzenetet a kívánt logikai állománynévvel (LFN-nel), és az állomány méretével. A kívánt LFN alapján az StM ellenőrzi, hogy a célutvonal létezik-e, illetve hogy létezik-e az a szülő könyvtár, ahova az állomány kerülne. Ezután az StM választ egy Storage Controller-t, ahol el szeretné helyezni az állomány fizikai példányát. Egy prepareToPut üzenetet küld az StC-nek az állomány méretével (itt már nem kell LFN-n, hiszen az StC-k nem tudnak a katalógusról vagy az egységes logikai névtérről), az StC erre válaszképp küld egy azonosítót, amellyel később el lehet érni az állományt (SURL), és egy URL-t, ahova fel kell tölteni az állományt (TURL-t). Az StM ezek után készít egy egyedi azonosítót az állományhoz (GUID-ot), majd elkészíti a megfelelő bejegyzést a katalógusban ehhez a GUID-hoz, amelyben tárolja az állomány méretét, és az elérés módját (azaz az SURL-t). Hogy a logikai névtérbe bekerüljön ez az állomány, a szülőkönyvtárban is létre kell hozni egy bejegyzést, az állomány nevével (azaz a kívánt LFN utolsó tagjával) és a GUID-jával. Miután ez megtörtént, az StM visszaadja a felhasználónak a TURL-t, amelyre az feltöltheti az állományt. Ez a folyamat látható a 3. ábrán.



3. ábra Állomány feltöltés folyamata

Hivatkozások

[1] The DataGrid Project, `http://eu-datagrid.web.cern.ch`

[2] EGEE, gLite, JRA1: Data Management, `http://egee-jra1-dm.web.cern.ch`

[3] Storage Resource Management, `http://sdm.lbl.gov/srm-wg`

[4] GridFTP, `http://www.globus.org/grid_software/data/gridftp.php`

[5] ByteIO, `https://forge.gridforum.org/projects/byteio-wg/`

[6] Grid Underground (GUG), `http://gug.grid.niif.hu/`

[7] NIIF Intézet, `http://www.niif.hu/`

[8] ClusterGrid, `http://www.clustergrid.niif.hu/`