ÁLLOMÁNYOK HATÉKONY SZINKRONIZÁLÁSA WEBSZOLGÁLTATÁSON KERESZTÜL

Dóbé Péter, dobe@inf.bme.hu
Dr. Szeberényi Imre, szebi@iit.bme.hu
Budapesti Műszaki és Gazdaságtudományi Egyetem
Irányítástechnika és Informatika Tanszék

1. Bevezetés

Különösen Grid rendszereknél, de más környezetben is felmerülhet az az igény, hogy egy távoli pontról frissítsünk fájlokat, amelyek két frissítés közötti időtartamban csak kis mértékben változnak. Ha a fájlok nagyméretűek, a teljes fájlt nem hatékony minden alkalommal a hálózaton átküldeni. Ilyen feladatra találták ki az rsync protokollt, melynek lényege, hogy csak a régi és új változat közötti különbségeket küldjük el.

A BME Irányítástechnika és Informatika Tanszéken kifejlesztett Saleve rendszer olyan ún. parameter study feladatok Grid környezetben való megoldását támogatja, amelyek nagyméretű állományokat állítanak elő. A rendszer az adatok átvitelére webszolgáltatást használ. A cikk egy olyan plugint ismertet a Saleve rendszerhez, amely rsync protokoll segítségével optimalizált fájlcserét végez webszolgáltatáson keresztül.

A cikk röviden bemutatja a Saleve rendszert és a rendszerhez kifejlesztett plugin konkrét megvalósítását egy tesztkörnyezetben. A megoldás azonban nem csak a Saleve rendszerben alkalmazható, ezért a cikkben külön kitérünk annak általánosítására is.

A C nyelvű mintaimplementáció a webszolgáltatás interfész kialakításához a gSOAP toolkit-re, az rsync algoritmus használatához pedig a librsync függvénykönyvtárra épül. A rendszer teszteléséhez a ClusterGrid környezetet használtuk.

2. Saleve rendszer

Parameter study vagy paraméter elemző feladatoknak nevezzük azokat a problémákat, amelyekben ugyanazt az algoritmust több százszor, esetleg több ezerszer különböző paraméterekkel, ill. paraméter tartománnyal kell lefuttatni. Jellemző az ilyen feladatokra, hogy az egyes futások között nincs csatolás, amiből az következik, hogy azokat időben tetszőleges sorrendben, vagy akár párhuzamosan is lehet futtatni. A végeredmény a futások eredményeiből, mint részeredményekből akkor állítható össze, amikor minden futás befejeződött.

Gyakorlati alkalmazások között igen sok ilyen probléma van, többek között a részecskefizikai számítások nagy része is ilyen, így akár egy függvény integráljának nagyon nagy pontosságú numerikus közelítése is kiszámítható lenne. Ha pl. felosztjuk az integrálandó értelmezési tartományt nagyjából egyenlő hosszúságú intervallumokra,

akkor az egyes intervallumokban az integrálközelítő összegek számítása külön-külön végezhető. Az így kapott részeredményekből pedig a végeredmény egyszerűen előállítható.

A gyakorlatban előforduló paraméter elemző feladatok számítási igénye igen nagy, ezért kézenfekvő az az igény, hogy az egyes futtatásokat időben egyszerre, több gépen végezzék el. Ez könnyen megtehető, mivel az egyes futások között nincs kapcsolat, csupán az egyes részek paraméterezéséről és a végeredmények begyűjtéséről kell gondoskodni. Az ilyen alkalmazások futtatására különösen alkalmasak a Grid rendszerek.

A részfeladatok szétosztása, a Grid rendszerbe való bejuttatása, elindítása és az eredmények begyűjtése a konkrét problémától jól leválasztható, tipikus feladat, melyet nem célszerű az adott szakmai feladatot megvalósító, többnyire nem informatikus szakemberre terhelni. Célszerű ezt a tevékenységtípust valamilyen módon támogatni.

A paraméter elemző alkalmazások nagy része az input adatokat fájlokból nyerik és az részeredményeket is fájlokba írják, melyeket a futás legvégén össze kell gyűjteni, esetleg össze kell másolni. Az ilyen jellegű alkalmazások fejlesztésének ill. a már meglevő programok Grid képessé tételét támogatja a tanszéken kifejlesztett *Saleve* rendszer[1] [2].

A Saleve függvénykönyvtár támogatja a már meglévő paraméter elemző alkalmazások átírását, és az új alkalmazások fejlesztését is, mégpedig úgy, hogy a fejlesztés eredményeként előálló futtaható alkalmazás további változtatás és újrafordítás nélkül egyaránt alkalmas lokális, vagy távoli futtatásra.

Az alkalmazás felhasználója szempontjából ez azt jelenti, hogy egyetlen egy bináris kód van, amit adott esetben egy asztali gépen, vagy notebook-on elindítva egymás után hajtja végre a programrészeket. Megfelelő paraméterezéssel indítva azonban képes egy távoli Saleve szerveren keresztül Gird, vagy cluster környezetben is elindítani az alkalmazást időben párhuzamosan több gépen. Ebben az esetben a Saleve szerver fogadja a feladatot, és elvégzi az elosztást, eltakarva a sokféle, különbözőképpen használandó Grid middleware technológiákat a felhasználó elől, majd az eredményt visszaküldi. A kommunikáció az alkalmazás és a szerver között webszolgáltatás alapokon történik. Szerver hiányában a feladatok szétosztása (többprocesszoros) helyi gépen is végezhető.

Az alkalmazás fejlesztője szempontjából csak néhány (2-3) függvény megtanulására és alkalmazására van szükség, melyet többnyire csupán a főprogramba kell beépíteni.

3. Webszolgáltatások, SOAP

A hálózat felett együttműködő gépek közötti adatcsere céljára manapság elterjedt mód a webszolgáltatások kialakítása. Ezekkel különböző platformok közötti adatcserére, eljáráshívásra nyílik lehetőség. A webszolgáltatás a HTTP-re épül, így egyrészt kihasználja egy már meglévő, nyílt szabvány előnyeit, másrészt a tűzfalakon sem igényel speciális beállításokat. A szolgáltatás igénybevételének módját egy gép által feldolgozható formátumban, WSDL-ben rögzítik [3].

Az adatok cseréje SOAP (Simple Object Access Protocol) [4], egy XML alapú protokoll segítségével történik. A SOAP felelős az adatok sorosított, hálózaton átküldhető formátumba történő átalakításáért. Ilyen adat lehet pl. egy távoli eljáráshívás, és annak paraméterei is. Ezeket a SOAP bizonyos kódolási szabályok alapján meghatározott XML formátumba alakítja. Az XML előnye többek között az, hogy – bár gépi feldolgozásra szolgál – szöveges formátumú, szükség esetén tehát emberi értelmezése, szerkesztése is

lehetséges, hátránya viszont, hogy nem a legtömörebb adatreprezentációt adja, ezért a feldolgozása erőforrás igényes lehet. Az XML gyökérelem itt a *SOAP boríték*, a paraméterek illetve visszatérési érték ennek egy-egy gyermeke. Az adatok típusa egyaránt lehet skalár (például egész), vagy összetett struktúra. Utóbbi esetben a struktúra egyes mezőit külön XML elemekben tároljuk.

4. Az rsync protokoll, alapelvek

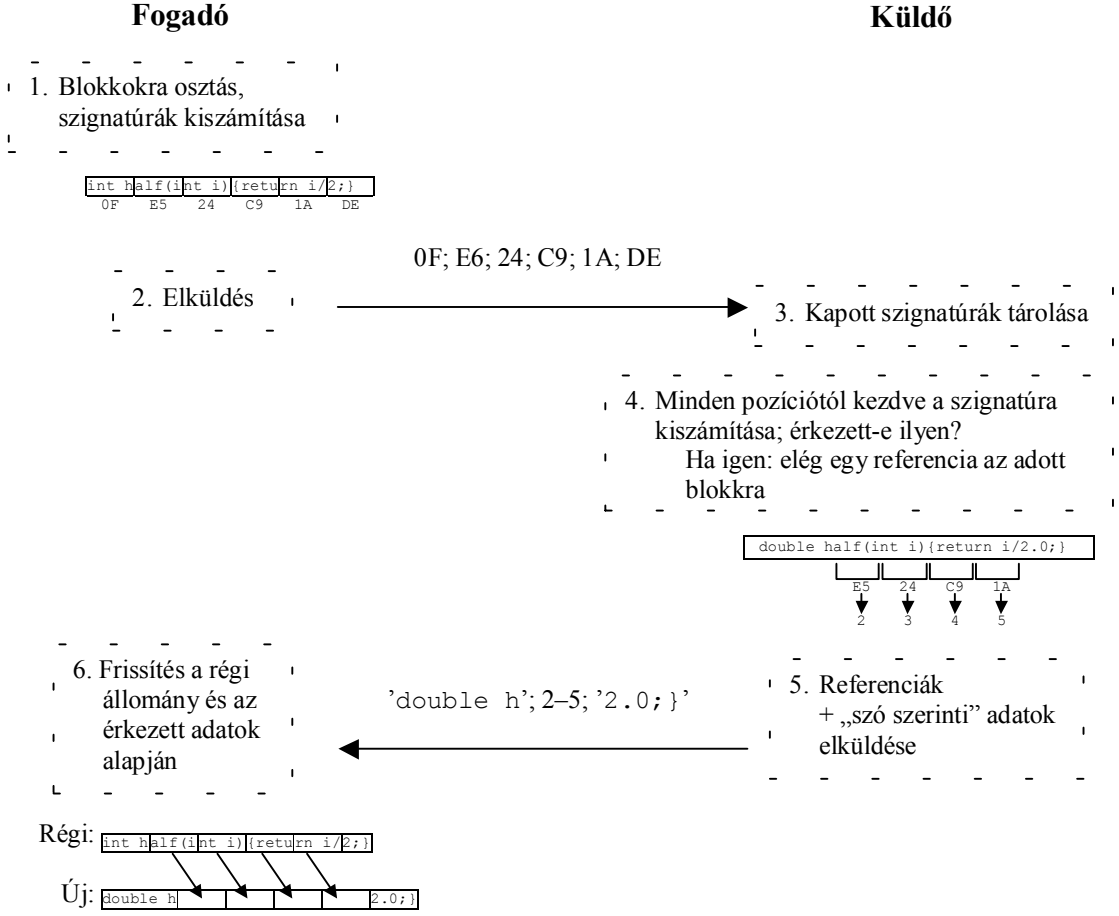
Gyakran előforduló probléma, hogy nagy méretű állományokat csak azért kell újra és újra teljes egészében átvinni az egyik gépről a másikra, mert tartalmuk kis mértékben, de megváltozott. Legjobb példája ennek amikor teljes naplóállományokat, vagy dokumentációkat mozgatunk csak azért, mert azok kis mértékben megváltoztak.

Lényegesen gyorsabban átvihető lenne az információ, ha a teljes állomány átvitele helyett az állománynak csak azon részeit vinnénk át, amelyek változtak, feltéve, hogy a változás az állománynak csak kis részét érinti.

Kis változások esetén jól alkalmazható az *rsync* [5] protokoll, melyet hálózaton összekapcsolt gépek közötti állománycseréhez fejlesztettek ki. Fontos megjegyezni, hogy a protokoll csak akkor alkalmazható, ha tudjuk, hogy mely oldalon keletkeztek az új adatok, és azokat hova kell átvinni, azaz a szinkronizálás nem szimmetrikus. Az rsync algoritmus hálózaton történő végrehajtásához kétirányú kapcsolatra van szükség. A továbbiakban *küldő*nek nevezzük azt a résztvevőt, akinél az állomány újabb változata van, *fogadó*nak pedig azt, aki ez alapján szinkronizál. A fogadó ill. küldő programok indítását a gyakorlatban daemon programokkal szokás segíteni, de számunkra ez most lényegtelen.

A kommunikáció kezdetekor a fogadó fél egyenlő hosszúságú blokkokra feldarabolja a saját (elavult) állományát, és blokkonként előállít egy-egy kivonatot (hash leképezést). Ezeket eljuttatja a küldőhöz, amely szintén elvégzi a kivonatolást a saját (friss) állományán az összes lehetséges módon feldarabolva azt, és megvizsgálja hogy van-e olyan kivonat, amely egyezik a küldő által küldött kivonat valamelyikével. Amennyiben az állomány valamely darabján van egyezés, akkor ezt a részt nem szükséges elküldenie a fogadónak, hiszen az a fogadónál már megvan; és nem változott meg. Csak azokat a részeket kell elküldeni, amelyeknek megfelelő kivonat nem érkezett a fogadótól. A fogadó ezek segítségével elő tudja állítani a frissített állományt: nagyobb részben a régi változatból másolással, kis részben a küldőtől érkező differenciából.

A kivonatok képzése és ellenőrzése valójában több szinten történik, ugyanis egy kivonatról (hash) csak adott valószínűség mellett tudjuk azt mondani, hogy nem származhat más adatból. A bonyolultabb és hosszabb kivonatok esetén annak valószínűsége, hogy az adat úgy változzon meg, hogy a kivonat közben azonos maradjon egyre kisebb. A hosszabb kivonatok átviteléhez azonban több adatra van szükség. Ezért az *rsync* először rövidebb kivonatokat készít. Ha ezek megváltoztak, akkor az adott állománydarab biztosan változott. Ha a rövidebb (gyengébb) kivonatok egyeznek, akkor az adott állományrészhez újabb, erősebb kivonat készül, mely alapján nagyobb valószínűséggel megállapítható a változás, vagy változatlanság ténye. Az 1. ábra az algoritmus egyszerűsített működését szemlélteti.



1. ábra: *Az rsync működése, öt bájtos blokkméret, egy bájtos szignatúra esetén*

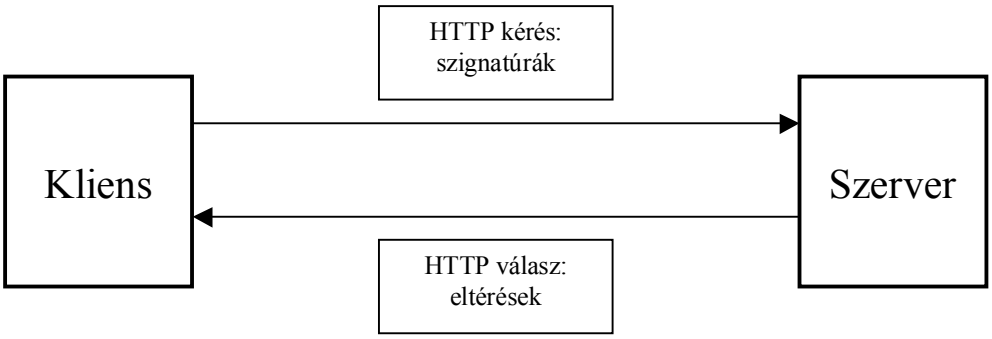
Az rsync algoritmussal nagyon nagy méretű és méretéhez képest jelentéktelen mértékben változó állományok frissítéséhez gyakorlatilag csak a kivonat átküldésére van szükség, amelyek nagyságrendekkel kisebbek a hozzájuk tartozó adatoknál. Akkor sincs jelentős forgalomigény, ha hosszabb szakaszok felcserélődtek vagy törlődtek az állományból, mert a protokoll biztosítja ezek felismerését a küldő oldalon amely értesíti erről a fogadó oldalt. Csak akkor nem működik hatékonyan ez az algoritmus, ha a kis változás a teljes állományon szétszórtan jelentkezik (például közel minden blokkban megváltozik egy-egy bájt), de ebben az esetben is helyes eredményt kapunk.

Az rsync algoritmust felhasználva olyan webszolgáltatást hoztunk létre, mely hatékonyan tudja csökkenteni az átviendő adatok mennyiségét és ezzel az átvitelhez szükséges időt. Célunk volt, hogy ez a lehetőség közvetlenül beépíthető legyen a Saleve programcsomagba, mely szintén webszolgáltatást használ.

5. Webszolgáltatás alapú fájlszinkronizálás megvalósítása

A megvalósítás szabványos C nyelven történt. Webszolgáltatások létrehozására és használatára nincs beépített nyelvi eszköz a C nyelvben, viszont a gSOAP toolkit [6] jól alkalmazható erre a célra. A gSOAP-ot használva a webszolgáltatások interfészét C vagy C++ függvényprototípus-deklarációk formájában írhatjuk le (egy header állományban). Ezek paramétereként gyakorlatilag tetszőleges C és C++ alaptípus, illetve saját összetett adatstruktúra használható. A szerver lehet különálló alkalmazás, de CGI-t is készíthetünk.

A feladatot egyszerűsíti az, hogy két üzenet elég a lefutásához: az elsőben a fogadó oldal közli a küldővel a kivonatokat, a másodikban a küldő adja az utasításokat (referenciákat és eltéréseket) az új állomány előállításához. Ezért lehetséges a HTTP feletti megvalósítás olyan formában, hogy a kliens a fogadó: a kérésben küldjük el a kivonatokat, a válaszban érkezik az eltérés. Mivel mindkettő gyakorlatilag tetszőleges hosszúságú, nem rögzített típusú adatfolyam, DIME (Direct Internet Message Encapsulation) [7] formátumban kódolja a SOAP.

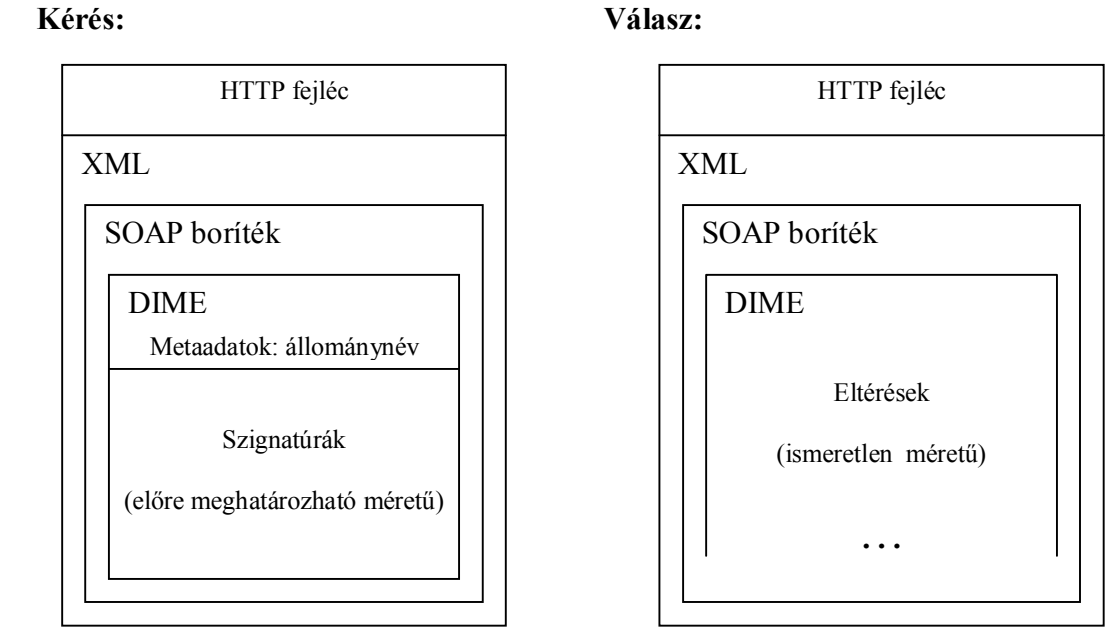


2. ábra: *Állomány frissítése a szerveren lévő változat alapján*

Sem a kivonatokat, sem az eltéréseket nem érdemes teljes egészében az operatív tárban elhelyezni elküldés előtt vagy érkezéskor, mivel egyrészt lehet, hogy nagy méretűek (különösen az eltérések), másrészt időveszteséggel jár ez a megoldás. Ehelyett lehetőségünk van adatfolyam (*streaming*) használatára, vagyis csak egy kis méretű puffert tartunk fenn, amelybe az éppen feldolgozott részadatot tesszük, és innen egyből elküldjük, ezáltal felszabadítjuk a puffert további adat számára.

A másik fél is hasonlóan, kisebb darabokban dolgozza fel az érkezett adatot. A HTTP támogatja a streaminget, viszont lehetnek bizonyos korlátozások. Például ha nem különálló szerveralkalmazást készítünk, a szerver oldalon követelmény lehet, hogy előre megadjuk a teljes HTTP üzenet hosszát, ellenkező esetben a webszerver egy „411 Length Required” hibával válaszol. A szignatúrákat tartalmazó üzenet hossza azonban az állomány méretének ismeretében előre meghatározható: ahány blokkra osztható fel, annyiszor kell venni a blokkonkénti kivonat hosszát; ehhez járul még egy rögzített hosszúságú fejléc.

Az eltérések teljes hossza nem jósolható meg sem a régi, sem a friss állomány méretéből, hiszen attól függ, hogy mennyi változás történt. Ez viszont nem jelent problémát, mert a HTTP válaszban a hossz előzetes megjelölése nem feltétlenül szükséges.



3. ábra: *A HTTP kérés és válasz felépítése*

Az rsync algoritmus végrehajtására legtöbbször az azonos nevű segédprogramot használják. Ez egy hálózaton működő program, amely ssh kapcsolat felett, vagy külön daemon folyamathoz kapcsolódva szinkronizál állományokat. Ennek megfelelően ez nem használható az általunk kitűzött célra. Egy másik eszköz az rdiff program, amellyel állományokon végezhetjük az rsync algoritmus egyes lépéseit külön-külön. Ennek a használata sem praktikus, mivel ez is egy kész alkalmazás, amelyet a saját programunknak kellene futtatni, és kommunikálni vele. A választott megoldás a librsync [8] függvénykönyvtár használata.

Az így létrehozott önálló, Saleve-től független kliens alkalmazás és szerver oldali CGI program hatékonynak bizonyult. A teljes állomány letöltéséhez képesti jelentős hálózatiforgalom-csökkenés természetesen némi számításigény-növekedéssel jár együtt. Ennek oka elsősorban a kivonatok intenzív számítása és összehasonlítása, kisebb mértékben az XML reprezentáció feldolgozása.

6. Összefoglalás

Az *rsync* protokollt felhasználva létrehoztunk egy olyan alkalmazás-interfészt, amely web protokollba épülve képes optimalizált adatátvitelre. A módszer különösen alkalmas nagyméretű, kis mértékben változó állományok hálózati adatforgalom szempontjából hatékony frissítésére.

A kidolgozott módszer alapján a tanszéken kifejlesztett Saleve rendszerhez megfelelő plugin modul készült, ami növeli a rendszer rugalmasságát, és felhasználási területét, mely különösen jól használható az ún. paraméter study feladatok megoldására Grid környezetben. A rendszer első teszteredményei bíztatóak.

Köszönetnyilvánítás

Jelen cikkben bemutatott munka az NKFP OM-00262/2004 és az EU INFSO-50883 (EGEE) [9] projekt támogatásával készült.

Hivatkozások

[1] Saleve: Simple Web-Services Based Environment for Parameter Study Applications, *IEEE Grid Computing Workshop*, 2005

[2] Saleve home page: http://gcsaleve.sourceforge.net/

[3] Web Services Description Language (WSDL) 1.1, *W3C Note 15 March 2001*, http://www.w3.org/TR/wsdl

[4] Simple Object Access Protocol (SOAP) 1.1, *W3C Note 08 May 2000*, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[5] A. Tridgell: Efficient Algorithms for Sorting and Synchronization, *Ph. D. Thesis, The Australian National University*, 1999

[6] gSOAP home page: http://www.cs.fsu.edu/~engelen/soap.html

[7] Direct Internet Message Encapsulation (DIME), *Internet Draft*, 2002, http://msdn.microsoft.com/library/en-us/dnglobspec/html/draft-nielsen-dime-02.txt

[8] librsync home page: http://librsync.sourceforge.net/

[9] Enabling Grids for E-Science: http://egee.ik.bme.hu, http://www.eu-egee.org