PHP a grid technológiában - egyszerűen és célratörően

Szalai Ferenc (szferi@niif.hu)

2004. január 15.

Kivonat

Ebben a cikkben összefoglaljuk az NIIFI ClusterGrid Infrastruktúrához készített web és PHP technológiát alkalmazó feladatütemező rendszer szoftver architektúráját és néhány implementációs részletét.

1. Bevezető

A grid technológia eredete az 1998-as évekig nyúlik vissza, noha az előzményei párhuzamos és elosztott rendszerek területén már korábban is megtalálhatóak voltak. Mára világossá vált, hogy a nagy teljesítményű adatfeldolgozást igénylő tudományos és műszaki számításoknak új informatikai infrastruktúra van szüksége. Ezt az infrastruktúrát megvalósító szoftver és hardver megoldásokat fogjuk az alábbiakban grid technológia néven illetni. Alább összefoglaljuk a hazai grid infrastruktúra fejlesztés egy prominens képviselőjét az NIIFI ClusterGrid Infrastruktúrát és azt, hogy az ehhez kapcsolódó szoftver fejlesztésekben, hogyan használtuk sikeresen a Web és a PHP technológia nyújtotta lehetőségeket.

2. Grid technológia alapjai

A grid technológia számtalan korábban meglévő ill. a jövőben elkészítendő szoftver és hardver elem együtteseként kínált informatikai infrastruktúra, melynek alapvető fogalma az erőforrás és a szolgáltatás.

A grid infrastruktúrában különféle erőforrások jelenhetnek meg egyszerre (számítási azaz cpu, tároló kapacitás azaz diszk, hálózati erőforrás stb.) ezáltal egy heterogén rendszer jön létre. A heterogenitás, mind a választott operációs rendszer, mind az azokon megtalálható szoftverek terén jelentkezik. A rendszerben található minden erőforrás egynéni felügyelet alá tartozik, tehát nincs központosított felügyelet. Ebből következően grid infrastruktúrát kiszolgáló szoftver elemeknek is a központosítás teljes hiányával kell rendelkezniük.

A rendszernek jellegzetes dinamizmust kölcsönöz, hogy az erőforrások száma és típusa időben is változhat.

Az erőforrások közős jellemzője, hogy őket valamilyen szoftver szolgáltatási közegen keresztül lehet elérni. Ezeknek a szolgáltatásoknak a halmaza képezi a grid infrastruktúra szoftver részét. Ezek a szolgáltatások szükségképpen szabványos protokollok segítségével kommunikálnak és Internet valamit bizonyos kiegészítő szabvá-

nyokat kell, hogy kövessenek. A jelenlegi szoftver megoldások közös jellemzője, hogy az alkalmazott szoftver megoldások általában nem szabványosak és kísérleti jellegűek.

Egy grid rendszer elosztottságának különféle mértéke lehet, jelenleg jellemzően úgynevezett nemzeti grid rendszerek vannak fejlesztés alatt. Egy ilyen nemzeti grid rendszerre kísértetiesen hasonlító kezdeményezés a későbbiekben bemutatandó NIIFI [1] által koordinált ClusterGrid Infrastruktúra projekt is. Később várható, hogy ezek a nemzeti grid infrastruktúra kezdeményezések összekapcsolódnak. Ez elsősorban az Európai Unión belül várható viharos gyorsasággal, ahol a hálózati infrastruktúra leginkább alkalmas rá.

Jelenleg a grid szerű rendszerek legnagyobb felhasználói bázisa az akadémiai kutató közösség, ahol a tudományos és műszaki szimulációk számítás igénye napról napra rohamosan nő. A grid rendszerek szoftver környezetének is elsősorban ezekhez a felhasználói igényekhez kell igazodniuk.

3. NIIFI ClusterGrid Infrastruktúra

Elmondható, hogy hatékony grid szoftver infrastruktúrát fejleszteni csak már kialakított hardver infrastruktúrához lehet. Az általános megoldások jelenleg még váratnak magukra.

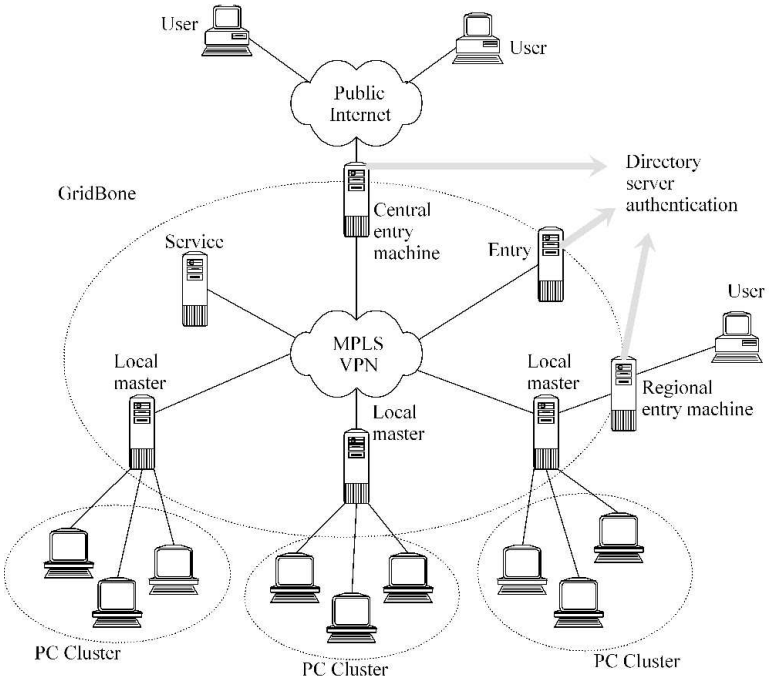
Magyarországon 2002 júliusában merült fel, hogy az Oktatási Minisztérium által a felsőoktatási intézmények számára PC laborok (34 labor kb. 2000 gép) beszerzésre kiírt pályázatot szerencsésen lehetne kombinálni a hazai grid szerű infrastruktúra hardver alapjainak megteremtésével. Az elképzelés szerint az egyetemeknek, főiskoláknak szállított PC laborok napközben szokásos feladatukat látják el, de éjszaka és hétvégénként egy egységes rendszerbe kapcsolódnak, és tudományos műszaki számításokban vesznek részt.

Világosan látnunk kell, hogy az így kialakítandó rendszer messze nem rendelkezik a grid infrastruktúra összes kívánalmával, hiszen legalábbis erősen homogén rendszerről van szó, de mindenképpen úgy gondoltuk az adott körülmények között ezen a rendszeren lehet elkezdeni a hazai grid technológia fejlesztéseket.

Az elmúlt másfél évben elsősorban a hálózati és operációs rendszer szinten történtek intenzív fejlesztések. Már a második generációnál tartunk. Hálózati szinten messzemenően támaszkodtunk az akadémiai hálózat gerincének (HBONE) lehetőségeire. Így fi gyelembe véve a biztonsági és megbízhatósági követelményeket egy teljesen szeparált MPLS VPN rendszer lett kialakítva az akadémiai gerinchálózaton belül. Ez lényegében egy privát mini internet, hiszen a útvonal választástól a DNS kiszolgálókon keresztül a felhasználó kezelésig minden privát szervereken és hálózaton zajlik. A rendszernek néhány jól védett belépési pontja van ahol a felhasználók "érintkeznek" a rendszerrel publikus hálózaton keresztül.

Operációs rendszer szinten egy Linux alapú vékony-kliens megoldás mellett döntöttünk és minden PC labort egy önálló klaszterbe szerveztünk egy vagy több központi szerverrel. A nappali "műszak" véget értével a gépek ezzel a rendszerrel indulnak és más hálózati szegmensbe kapcsolódnak. Ezzel sikerült nem csak hálózati szinten, de operációs rendszer szinten is szeparálni a ClusterGrid rendszert, minden más rendszertől.

A ClusterGrid egy országos összefogáson alapuló rendszer szinte minden a rendszerbe kapcsolt egyetem és főiskola tevékenyen részt vesz a hálózati, szoftver fejlesztésekben vagy a tesztelésben és az üzemeltetésben.



1. ábra. NIIFI ClusterGrid Infrastruktúra [8]

4. ClusterGrid Bróker

Az első generációs ClusterGrid rendszer a Condor [4] nevű feladatütemező (bróker) rendszert használta látványos kudarccal. Ennek oka elsősorban annak köszönhető, hogy minden küzdelem és fejlesztés ellenére a Condor mégiscsak egy klaszterekre fejlesztett feladatütemező és ebben az elosztott környezetben mar nem volt képes hatékonyan és biztonságosan működni. A problémára jól rámutat az a 2200%-os overhead ami egyes programok futási idejében jelentkezett.

Ekkor döntöttünk a mellett, hogy saját feladatütemezőt készítünk, mely fi gyelembe veszi a ClusterGrid rendszer sajátosságait, de egyben prototípusa is egy olyan erőforrás ütemezőnek, amit a ClusterGrid rendszer keretein kívül is működőképesnek gondolunk.

4.1. PHP és Web technológia - miért?

Az erőforrás ütemező elkészítésekor a web technológia mellett tettük le voksunkat. Ennek a döntésnek a magyarázata több okra vezethető vissza:

1. Figyelembe kellett vennünk, hogy jelenleg a grid technológia területén nincsenek széles körben elterjedt szabványok, továbbá, hogy a jelenleg rendelkezésre álló szoftver eszközök (elsősorban a Globus [3] és Condor és hasonló rendszerek) production környezetben használhatatlannak bizonyultak, mind biztonsági, mind megbízhatósági, mind üzemeltethetőségi szempontból.

2. Figyelembe kellett vennünk, hogy a grid technológia egy igen gyorsan változó

kutatási, fejlesztési terület, mind a felhasználók, mind az üzemeltetők oldaláról. Ezért gyorsan kell tudunk reagálni ezekre a változásokra.

3. A web technológián belül számtalan a grid technológián belül is felmerülő kérdésre már kiforrott, hatékony és megbízható megoldás található: adatbázis hozzáférés, autentikáció nyilvános kulcsú technológiával, felhasználói interfészek stb.

4. Jelenleg a web technológia alapjain kifejlődni látszó web szolgáltatásokban látják többen a grid technológia szoftver környezetének alapjait. Mivel ezek igen közel állnak technológiai szempontból egymáshoz, feltehetőleg könnyen tudunk mi is lépni ebbe az irányba, ha szükséges lesz.

5. A web technológián belül - mint sokan másoknak, így nekünk is a - a PHP-vel volt a legnagyobb tapasztalatunk.

4.2. Architektúra

A feladatütemezőtől azt várjuk, hogy a felhasználók által indított feladatokat hatékonyan a rendelkezésre álló erőforrásokon futtassa, a keletkezet eredményeket elérhetővé tegye. Továbbá a feladatok futásáról információkat szolgáltasson a felhasználóknak.

4.2.1. Mi a feladat?

A grid rendszerek ezen belül is az ütemezők készítésnél az egyik legfontosabb feladat, hogy defi niáljuk mi is a rendszeren végrehajtandó elemi feladat és milyen formában áll az az ütemező rendelkezésére. Számos - elsősorban klaszter - ütemező tanulmányozása után arra a megállapításra juthatunk, hogy a legkényelmesebb az lenne, ha a feladatok globálisan egyedi azonosítóval rendelkeznének és egy elemi feladatot egy könyvtár struktúra defi niálná a fi lerendszeren. Az ötlet nem új keletű, már az OpenStep [7] specifi káció egyes elemeinek UNIX rendszereken történő implantációjakor is felmerült és a GnuStep [5] is így kezeli az alkalmazásokat. E mellet döntöttünk mi is.

A ClusterGrid rendszerben egy feladatot egy könyvtár reprezentál, melynek meghatározott struktúrával kell rendelkeznie. Es a struktúra gyakorlatilag megegyezik a FHS-ben [2] rögzített alapelvekkel. E mellet a könyvtár struktúrának tartalmaznia kell egy ún. *submit* állomány, ami a feladat specifi kációját tartalmazza. Ez magában foglalja a futtatandó bináris állomány nevét, a szükséges paramétereket, környezeti változók beállítását, a feladat típusának (batch, szekvenciális, pvm vagy mpi feladat) meghatározását stb. A specifi káció formátuma jelenleg egyszerű szöveges állomány melyben név, érték párok szerepelnek (kísértetiesen emlékeztet egy php.ini állományra).

4.2.2. Komponensek, objektumok, szolgáltatások

A feladatütemező rendszer az alábbi komponensekből áll:

1. **Submit**: a ClusterGrid rendszer belépési pontjain található. A felhasználók ezen keresztül indítják feladataikat.

2. **Exec**: a ClusterGrid rendszer minden klaszter erőforrásának frontend gépen található. Ez biztosítja az interfészt a Submit komponens és az erőforrások között.

3. **Interfész a helyi ütemezőhöz**: egy klaszteren az ütemezést nem magunk végezzük, hanem rábízzuk a már létező klaszter ütemező rendszerek valameikére (Condor, SGE, stb.). Ez a komponens biztosít egy egységes interfészt a különféle klaszter ütemezők és a Exec komponens között.

4. **Biztonságos futtatókörnyezet**: a helyi ütemezőn keresztül biztosítja a feladatok biztonságos végrehajtását.

A fenti komponensek legnagyobb része megvalósítható, mint tényleges grid szolgáltatások együttműködésének eredménye. Hiszen például a *Submit* komponensnek szüksége van az erőforrásokról információkra, kezelnie kell a feladatokat és azok állapotait, valamint ütemeznie kell a feladatokat erőforrásokra. Ezek a feladatok néhány szolgáltatás (információs rendszer szolgáltatás, ütemező szolgáltatás, kommunikációs szolgáltatás stb.) segítségével könnyen megvalósíthatóak.

A rendszer a szolgáltatásokat, mint objektumokat valósítja meg - amennyire ez PHP-ban lehetséges. Ezek a következők:

- **Kommunikációs alrendszer**: a kommunikációt HTTPS protokoll felett valósítja meg.
- **Információs Rendszer**: A rendszerben található erőforrásokkal kapcsolatos információk begyűjtését, tárolását végzi és elérhetővé teszi más szolgáltatások számára.
- **Tranzakció kezelés**: Az erőforrásokon végrehajtandó műveletek tranzakciók formájában jelennek meg.
- **Notification rendszer**: A erőforrások és az ütemezők bizonyos események bekövetkezéskor notifi cation-okat küldenek egymásnak.
- **Ütemező rendszer**: A feladatok erőforrásokhoz rendelését végzi valamint fi gyelemmel kíséri a feladatok állapot változásit az erőforrásokon.
- **Egyéb objektumok**: a hibakezelést, loggolást, adatbázis hozzáférést biztosító objektumok.

Az objektumok relációs adatbázisban tárolnak minden perzisztens adatot állapotukról: a feladatok leírását, állapotukat, az erőforrások állapotát, az adott erőforráson végrehajtandó tranzakciókat stb. Az alábbiakban valamivel részletesebben bemutatjuk a három legfontosabb komponens ill. objektum szerkezetét.

4.3. Kommunikáció

Az ütemező rendszernek a felhasználokkal parancssori és web-es interfészen keresztül is képesnek kell lennie kommunikálni. E mellett a *Submit* és *Exec* komponensek között is hálózati kommunikációra van szükség. Mindkét esetben egy absztrakt kommunikációs csatornát hozunk létre. Ennek a kommunikációs csatornának a jelenlegi implementációja HTTPS feletti POST üzenetváltás. Ezt a PHP curl interfészével könnyedén és egyszerűen el lehetett készíteni. Fontos megjegyezni, hogy mind a *Submit* és *Exec* komponensek, mind pedig a *Submit* komponens és felhasználói interfész között nyilvános kulcsú titkosítást alkalmazunk oly módon, hogy mind a szerver, mint a kliens oldalon ellenőrizzük a jogosítványok hitelességét. Ezért a ClusterGrid rendszer saját CA szolgáltatással rendelkezik, mely a közeljövőben a NIIFI országos CA szolgáltatása vált majd fel.

4.4. Tranzakció és notification rendszer

A rendszer tervezésekor fontos szempont volt a hibatűrés. Ezért döntöttünk úgy, hogy a mellett, hogy az objektumok perzisztens adatit adatbázisban tároljuk a *Submit* és *Exec* komponensek között a feladatokat tranzakciós formában továbbítjuk. A rendszeren a feladatokkal végezhető műveletek egy az egyben tranzakciókra képeződnek le:

- **submit tranzakció**: feladat végrehajtás kezdeményezése.
- **remove tranzakció**: a feladat végrehajtás befejezése. A feladatot és a hozzá tartozó állományokat ill. adatbázis bejegyzéseket minden bróker komponensből törölni kell. A feladat akkor törlődik véglegesen, ha ez a tranzakció sikerrel végrehajtódik.
- **result tranzakció**: a feladat által létrehozott eredmények begyűjtése. A feladat eredményei az erőforrásokon keletkeznek. Ezeket az állományokat kell a felhasználó által kívánt helyre eljuttatni. A feladat eredményeinek azokat az állományokat tekintjük, amik a feladat adott erőforráson való elindulása után jöttek létre vagy módosultak. A feladat eredményeit a rendszer általános esetben egy tar.gz állományban prezentálja.
- **status tranzakció**: a feladat aktuális állapotának meghatározása. A feladat végrehajtás kezdeményezése után, a feladat számtalan állapotváltozáson megy át. Ezekről a változásokról, valamint az esetleg bekövetkező hibákról a felhasználót tájékoztatni kell. A feladatok állapotai megfelelnek a DRMAA [6] specifikációban rögzítetteknek.

Ha a rendszerben új tranzakció jelenik meg akkor arról a megfelelő erőforrást egy notification-al értesítjük, azért hogy elkerüljük a *Submit* komponensek folyamatos pollolását. E mellett természetesen a *Exec* komponensek időnkét érdeklődnek a *Submit* komponenseknél a végrehajtandó tranzakciókról, annak érdekében, hogy a valamilyen meghibásodás miatt félbemaradt vagy hibás tranzakciókat befejezhessék.

4.5. Biztonságos futtató környezet

A bróker tervezésekor fontos szempont volt a biztonságos feladatvégrehajtás megvalósítása. Korábbi tapasztalatokból okulva mindenképpen el akartuk kerülni, hogy a felhasználók közvetlenül hozzáférhessenek az egyes klaszterekhez, ill., hogy bármiféle felhasználói adat nyilvántartásra szükség legye az erőforrásokon. Ugyanakkor az is követelményként fogalmazódott meg, hogy a feladatokat egymástól is védeni kell.

Ezt úgy oldottuk meg, hogy minden erőforráson minden feladat önálló felhasználói és csoport azonosító alatt fut. Minden feladathoz annak a rendszerből való törléséig átmenetileg és dinamikusan rendelődig ilyen felhasználói és csoport azonosító pár. Továbbá azok a tranzakciók melyek file műveleteket is végeznek azok ezekkel a a feladathoz rendelt azonosító pár alatt teszik ezeket.

A dinamikus azonosító allokálását a Grid Underground [9] projekt keretében létrehozott *libnss-dynmap* és *IDRegister* szoftverek segítségével valósítottuk meg.

5. Összefoglalás

Összefoglalva elmondhatjuk, hogy a ClusterGrid Infrastruktúra feladatütemező rendszerének megvalósításakor a web és PHP technológia nyújtotta keretek kielégítőnek

és hatékonynak bizonyultak. A fejlesztés folyamatosan zajlik új ötletek merülnek fel, amiket gyorsan tudunk megvalósítani a kialakított keretek között. A jelenlegi rendszer legnagyobb erénye meglátásunk szerint a kicsi áttekinthető és egyszerű szerkezete.

Meg kell említenünk, hogy a fejlesztés a ClusterGrid üzemeltetői közösségének aktív bevonásával folytatódik a jövőben nyílt forráskódú fejlesztések alapelvei szerint.

A jövőben a legnagyobb változást az ütemező struktúrájában, más grid rendszerekkel való összekapcsolása hozhat, melynek alapjain jelenleg dolgozunk.

Hivatkozások

[1] http://www.niif.hu/

[2] http://www.pathname.com/fhs/

[3] http://www.globus.org/

[4] http://www.cs.wisc.edu/condor/

[5] http://www.gnustep.org/

[6] http://www.drmaa.org/

[7] http://www.gnustep.org/resources/OpenStepSpec/OpenStepSpec.html

[8] http://www.clustergrid.iif.hu/

[9] http://gug.sf.net/