Grid rendszerek és alkalmazások

kacsuk@sztaki.hu

A grid rendszerek fejlődésében három hullámot különböztethetünk meg. Az első hullámnak tekinthető tudományos gridek példái az európai EGEE grid rendszer [1], illetve ennek magyarországi verziója, a HunGrid [2], melyben eredetileg a SZTAKI és az RMKI játszották a fő szerepet, de mára már a BME, az ELTE és az NIIFI is biztosít hozzá erőforrásokat.

Az EGEE projektet és a HunGridet mutatja be célszámunk első cikke, amit *Kárász Edit* írt. A HunGrid mellett létre jött a magyar ClusterGrid [3] is az NIIF gondozásában. Egy másik jelentős nagy európai projekt a SEE-GRID [4], amely a délkelet-európai országok közös gridrendszerét kívánja létrehozni. Erről a projektről és eredményeiről számol be *Kozlovszky Miklós és szerzőtársainak* cikke. Ezek a gridek ma már stabil szolgáltatást nyújtanak, de csak korlátozott számú PC bevonására képesek. A két fent említett magyar tudományos gridben például együttesen mintegy 1700 PC van összekötve.

A második hullám a vállalati gridrendszerek kiépítését jelenti. A világszerte megfigyelhető tendenciának megfelelően hazánkban is megjelent az első vállalat (AMRI), amely elkezdte saját belső gridrendszerének kiépítését és az első hálózati szolgáltató (Econet), amely grides szolgáltatás kiépítésén dolgozik. A harmadik hullám a szolgáltatói (közmű) jelleg megjelenése lesz. Ez a hullám még nem kezdődött el, az ehhez szükséges K+F projektek jelenleg alakulnak, vagy indulnak szerte a világban.

A tudományos számításokat támogató gridrendszerek eredeti törekvése az volt, hogy bárki erőforrást tudjon felajánlani a grid rendszerek számára, illetve erőforrást tudjon onnan szerezni dinamikusan, igény szerint, minél több erőforrás bevonásával. Sajnos az eredeti célkitűzés nem valósult meg teljes mértékben. Jelenleg két egymástól jelentős mértékben eltérő irányzat figyelhető meg a tudományos grid rendszerek fejlődésében.

Az első irányzat arra törekszik, hogy olyan grid szolgáltatást hozzon létre, ami a felhasználók számára stabilan, szolgáltatásként érhető el. Egy erőforrás grid erőforrássá alakítását grid-szoftverrendszerek feltelepítésével érik el, amelyek bonyolultak, telepítésük és karbantartásuk komoly szakértelmet igényel. Ennek eredményeképpen a grid erőforrásait nem egyének, hanem intézmények biztosítják, amelyek professzionális rendszergazdákat alkalmaznak és ezzel garantálják a grid magas színtű rendelkezésre állását. Ilyen rendszerekre példa a legnagyobb európai grid, az EGEE Grid (és annak magyar verziója a HunGrid), vagy az Egyesült Királyságban alkalmazott Nemzeti Grid Szolgáltatás [5], illetve a magyar ClusterGrid, amit az NIIF üzemeltet. Az eredeti célkitűzésből tehát nem teljesül az a kitétel, hogy bárki erőforrást biztosíthat a grid számára, így ezért a grid erőforrások száma meglehetősen korlátozott. (A világ legnagyobb ilyen gridjében, az EGEE Gridben is csak körülbelül 40 ezer PC van összekötve.) Ugyanakkor bárki használhatja az ilyen gridek erőforrásait, ha sikeresen beregisztrált egy adott grid engedélyezési hatóságánál és onnan egy grid-jogosítványt kapott. A jogosítványnak kiemelt szerepe van a tudományos gridek biztonságtechnikájában. Az ehhez kapcsolódó grid biztonsági kérdéseket részletezi *Kővári Kálmán* cikke.

A második irányzat, az úgynevezett desktop grid (DG) technológia, az első irányzatnak épp a komplemense az eredeti célkitűzések tekintetében. Ez az irányzat lehetővé teszi, hogy bárki erőforrást allokáljon a grid számára, de jelentősen korlátozza a gridet hasznosító felhasználók számát. Ennek az irányzatnak az egyik közismert példája a SETI Grid rendszer [6], ahol kb. 1,5 millió PC-t használnak. A SETI és több, az egész világra kiterjedő publikus DG projekt (EINSTEIN@home, LHC@home stb.) a BOINC technológiát [7] alkalmazza, akárcsak a SZTAKI Desktop Grid [8], melynek publikus verziója egy, az ELTE által kifejlesztett matematikai alkalmazás [9] végrehajtásába vont be több mint 27 ezer számítógépet a világ számos országából.

Ugyanakkor a publikus DG rendszereket nem használhatja akárki, hanem egy, vagy néhány nagy projekt számára biztosítanak extra számítási kapacitást. Azaz éppen azok, akik a PC-jüket felajánlják, nem profitálnak a rendszer használatából. Ezt az ellentmondást próbálja feloldani a Jedlik Ányos projekt [10], melynek keretében az Econet munkatársai olyan üzleti modellt próbálnak kidolgozni a desktop gridek alkalmazásában, ami lehetővé teszi a PC donorok jutalmazását is. Ugyancsak ennek a projektnek a keretében a SZTAKI olyan megoldást igyekszik kidolgozni, melynek segítségével a lokális desktop gridrendszerek hierarchiába szervezhetők. Az ehhez szükséges új ütemezési megoldásokat mutatja be *Farkas Zoltán* cikke.

A desktop gridrendszerek kutatásában és fejlesztésében két magyar intézmény is a nemzetközi élvonalban található. Az Econet kiemelten biztonságos globális desktop gridrendszere már a külföldi vásárlók figyelmét is felkeltette. A SZTAKI Desktop Grid hierarchikus verziója pedig a BOINC fejlesztők figyelmét vonta magára, kísérleti felállítása már két hazai vállalatnál is megtörtént, fogadtatása mindkét helyen igen kedvező volt. A SZTAKI

nemzetközi elismertségét mutatja ezen a területen, hogy a 2008 januárjában kezdődő EDGeS (Enabling Desktop Grids for e-Science) EU FP7 projektet koordinálja. Az EDGeS célja épp a fent említett kétféle grid rendszer integrálása egyetlen nagyteljesítményű rendszerbe [11].

A feladatok elosztása a gridben komoly kihívást jelent, mivel a különböző tudományos gridek egymástól szeparáltan működnek és nem képesek egy általánosan elfogadott szabvány mentén kölcsönös együttműködésre. Ennek a problémának a megoldására mutat példát *Kertész Attila* cikke, melyben a szerző egy gridek feletti Meta-Brókert mutat be, ami lehetővé teszi a feladatok elosztását nemcsak egy griden belül, hanem több grid között is.

A grid rendszerek elérésében fontos szerepet játszanak a grid portálok. A SEE-GRID hivatalos portálja a SZTAKI által kifejlesztett P-GRADE (lásd Kozlovszky és társainak cikkét), ami egy általános célú portál [12], magasszintű grafikus workflow alkalmazások gyors létrehozását és grides futtatását támogatja, elrejtve az aktuális grid specifikus tulajdonságait. A P-GRADE portál a legelterjedtebb grid technológiákat támogatja (GT2, GT4, LCG-2, gLite), sőt lehetővé teszi, hogy egy workflow alkalmazás különböző tevékenységei egyidejűleg több különböző típusú gridben is futhassanak [13]. Ezen az elven alapul a P-GRADE GIN VO Resource Test Portal [14], ami az OGF GIN (Grid Interoperability Now) munkacsoportja számára biztosít egy olyan megoldást, mellyel a legkülönbözőbb gridrendszerek erőforrásai egyidejűleg tesztelhetők. Ez a HunGrid hivatalos portálja is, amit az ELTE biztosít szervízszinten a magyar felhasználók számára [15]. *Pasztuhov Dániel és szerzőtársai* egy másik GridSphere-alapú [16] portált mutatnak be cikkükben, ami egy konkrét alkalmazás támogatására lett létrehozva a BME-n, ugyanakkor olyan eszközöket biztosít, mellyel más alkalmazások támogatásához is igazítható.

Alkalmazások nélkül a grid nem sokat érne, így a mi célszámunk sem lehetne teljes. *Kozlovszky Miklós és társai* két magyarországi példát is mutatnak, melyek a SEE-GRID projekt keretében lettek kifejlesztve. Az első egy e-piactér, a második egy mérnöki alkalmazás. *Pasztuhov Dániel és társai* pedig egy vasbeton hídgerendák térbeli alakváltozásainak számítására szolgáló grides alkalmazást írnak le. Az ilyen alkalmazások grides támogatására a BME a Saleve rendszert, a SZTAKI pedig a P-GRADE portált fejlesztette ki. A Saleve rendszert részletesebben *Dóbé Péter és szerzőtársainak* írása tárgyalja. A harmadik grid-alkalmazás, amit *Ádám Rita és Bencsik Attila* cikke mutat be, a meteorológiai változók és a levegőben jelenlévő pollenkoncentráció közötti kapcsolatot keresi és elemzi az allergiás betegek érdekében.

Célszámunk természetesen nem törekedhetett teljességre. Ezzel együtt reméljük, hogy a kedves olvasó ötleteket talál arra nézve, hogyan kezdheti el alkalmazni a magyar és nemzetközi grid rendszereket nagyszabású feladatainak megoldása érdekében.

Kacsuk Péter vendégszerkesztő
MTA SZTAKI

Irodalom

[1] EGEE: Enabling Grids for E-SciencE, http://public.eu-egee.org/

[2] HunGrid web lapja: http://www.lcg.kfki.hu/?hungrid&hungridgeneral_hun

[3] Magyar ClusterGrid web lapja: http://www.clustergrid.niif.hu/

[4] SEE-GRID projekt web lap: http://www.see-grid.org/

[5] Egyesült Királyság Nemzeti Grid web lapja: http://www.ngs.ac.uk/NGS-tacu.html

[6] D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer: SETI@home: An Experiment in Public-Resource Computing, Communications of the ACM, Vol. 45., No.11, November 2002, pp.56–61.

[7] D. P. Anderson: BOINC: A System for Public-Resource Computing and Storage. 5th IEEE/ACM Int. Workshop on Grid Computing, 8 November 2004, Pittsburgh, USA. http://boinc.berkeley.edu/grid_paper_04.pdf

[8] SZTAKI Desktop Grid: http://szdg.lpds.sztaki.hu/szdg, http://www.desktopgrid.hu

[9] Kornafeld, A., Kovács, A.: Szuperszámítógépes teljesítmény szuperszámítógép nélkül – A BinSYS projekt, Networkshop 2006, Miskolc, NIIF, 2006. http://www.lpds.sztaki.hu/~kadam/BinSYS.pdf

[10] "Új generációs grid technológiák kifejlesztése és meteorológiai alkalmazása a környezetvédelemben és az épületenergetikában" Jedlik Ányos projekt weblapja: http://www.hagrid.hu/

[11] Balaton. Z., Farkas Z., Kacsuk P., Kelley, I., Taylor, I., Kiss T: EDGeS: The Common Boundary Between Service and Desktop Grids, beküldve: CoreGrid Integration Workshop 2008.

[12] P-GRADE grid portál: http://portal.p-grade.hu/

[13] Peter Kacsuk, Tamas Kiss, Gergely Sipos: Solving the Grid Interoperability Problem by P-GRADE Portal at Workflow Level. Proc. of the 15th IEEE International Symposium on High Performance Distributed Computing (HPDC-15), Paris, 2006.

[14] P-GRADE GIN Resource testing: https://gin-portal.cpc.wmin.ac.uk:8080/gridsphere/gridsphere

[15] HunGrid portal: https://pgrade.inf.elte.hu/gridsphere/gridsphere

[16] M. Russell, J. Novotny, O. Wehrens, "GridSphere: An Advanced Portal Framework", GridSphere Project Website (www.gridsphere.org)

Magyarország az EGEE együttműködésben

KÁRÁSZ EDIT

KFKI Részecske- és Magfizikai Kutatóintézet
karasze@sunserv.kfki.hu

Lektorált

Kulcsszavak: grid, e-science, EGEE

Az EGEE2 (Enabling Grids for E-sciencE 2) célja egy nemzetközi kutatói grid kialakítása és fejlesztése. Magyarország évek óta szerepet vállal az infrastruktúra sejtjeit képező klaszterek, vagy úgynevezett állomások üzemeltetésével és felhasználói programok fejlesztésével. Cikkünkkel egy rövid áttekintést szeretnénk nyújtani a projekt céljairól, működési struktúrájáról, és a hazai szerepvállalásról.

1. Az EGEE bemutatása

Az EGEE – teljes nevén Enabling Grids for E-sciencE –, az egyik legnagyobb befektetéssel járó és legtöbb országot megmozgató Európai Uniós projekt [3]. A projekt célja, hogy alkalmassá tegye a grid technológiákat az E-kutatás számára. Ezzel a megfogalmazással rögtön két újabb fogalmat is nyertünk, az EGEE létrejöttének kulcsgondolatait; azaz mi az a grid és mit takar az E-science?

Manapság a kutatások szerves részét képezik a különböző számítástechnikai alkalmazásokra épülő tesztek, szimulációk, illetve az adatok jól szervezett tárolására, visszakeresésére és biztonságos megőrzésére kialakított struktúrák. A cél nem más, mint nemzetközi szinten összefogni a kutatásokat, és megosztani egymással a kutatási eredményeket. Ennek természetes közvetítő közege az internet. Interneten összekapcsolt adatbázisokkal és felhasználói szoftverek segítségével távoli kutatóintézetek munkái összefonódhatnak, nemzetközi együttműködések alakulhatnak ki. A mérési eredmények megosztásával lehetővé válik a párhuzamos kutatás ugyanazon eredmények alapján. E-science (E-tudomány) alatt azokat a nagy mennyiségű adattal dolgozó, nagy számításigényű és kiterjedt együttműködést kívánó kutatásokat értjük, amelyek csak hatalmas, területileg is elosztott hálózatokba kötött erőforrások segítségével végezhetők.

A grid szó számítógépes értelemben először Ian Foster és Carl Kesselman „The Grid: Blueprint for a new computing infrastructure” [1] című munkájában jelent meg az 1990-es évek elején. Azóta a szuper-számítástechnika egyik alapfogalmává nőtte ki magát, a klaszterek és az önálló szuperszámítógépek mellett. A markáns különbség a fizikai kiterjedésben lelhető: míg egy szuperszámítógép manapság néhány szekrény méretével ér fel, egy klaszter egy gépteremben elfér, addig egy kisebb grid összefogás is legalább országos méretű. Az esetek többségében a grid lokális gócait képező állomások (site) a nyilvános interneten keresztül kommunikálnak egymással, bár meg kell jegyezni, hogy egyre több a dedikált, illetve akadémiai hálózatra csatlakozó állomás a mai grid implementációkban.

Összeállítás szempontjából egy intézetnek olcsóbb létrehozni egy zárt klasztert, mint egy grid állomást. Míg az előbbi esetben csak a számolást végző munkagépekre (workernode) és az ütemező gépre (scheduler) van szükség, az utóbbi néhány extra funkciót is igényel, továbbá távolról valamilyen szinten mindenképpen elérhetőnek kell lennie. Míg egy zárt rendszerben nincs szükség túl erős biztonsági intézkedésekre, hiszen az akár a világhálóról leválasztva is működhet, a grid állomásokon többnyire szükség van egy biztonsági szakértőre is. A grid rendszergazda feladatköre is rendkívül szerteágazó, hiszen a világméretű rendszer fejlesztéseit naprakészen követnie kell, hogy az állomás gépein futó grid szolgáltató programcsomag (grid middleware) mindig a lehető legfrissebb legyen.

Az állomásnak nagyobb a létrehozási és a fenntartási költsége is, mint a hagyományos klaszterek esetében – ez utóbbiak kihasználtsága viszont korántsem optimális: a felhasználók köre sokkal szűkebb, és egyes időszakokban az erőforrások nincsenek kihasználva. A kompatibilis szolgáltató programcsomagot futtató állomások viszont képesek egymás között is elosztani a munkát. Ezt az elosztást külön erre a feladatra dedikált, úgynevezett erőforrás-elosztó (resource broker) gépek végzik. És mivel ez a szolgáltatás sem központosított, ezért gyakorlatilag bármelyik állomás kiesését képes elviselni a teljes rendszer, üzemzavar nélkül. Ennek az igazi értéke azok számára lehet nyilvánvaló, akik már jártak úgy, hogy határidő előtti utolsó napra lett meg végre minden adat a futtatáshoz és aznap éppen állt a helyi klaszter.

A globális mércének még egy nagy előnye van: ha én pillanatnyilag nem futtatok, de valaki a világ másik sarkában éppen most szeretne sokkal nagyobb munkát kiszámoltatni, mint amit az ő helyi kapacitása elbírna, nyugodtan használhatja az én üresen álló állomásomat is. Ez a modell nagy léptékben, sok felhasználó esetén nagyon szép, egyenletes viselkedést mutat. Így a kihasználtságot is figyelembe véve egy grid rendszer

megfelelően szervezve gazdaságosabb tud lenni, és sokkal nagyobb összkapacitást tud nyújtani, mint sok kisméretű klaszter.

Egy grid rendszer legnagyobb részét a számítási kapacitást képező munkagépek teszik ki. Ezek gyakorlatilag egy homogén szoftverkörnyezetet nyújtanak egy ingyenes vagy kommerciális operációs rendszer felett, amely a grid ütemező programjának kliens-oldalát futtatja. A felhasználó-kezelés már az operációs rendszer keretén belül történik, nincs külön azonosítási rendszer, egyedül a számítási vezérlő (Computing Element, CE) gép IP-címét kell ismerniük, hogy onnan fogadják a futtatási információkat, és oda küldjék a jelentéseket.

A számítási vezérlő a központi ütemező az állomáson, ez ellenőrzi az elküldött feladatok (job) tulajdonosának digitális személyazonosságát és jogosultságát az állomáson való futtatásra, küldi ki a gépekre a feladatokat terhelés szerint elosztva, összegyűjti a futtatások eredményeit és továbbítja kifelé, a megrendelőhöz.

A megrendelő ilyen értelemben az erőforrás elosztó. Ez a típus már nem szerves tartozéka egy kisebb állomásnak, elég néhány belőle több állomásnak. Feladata, hogy a felhasználói kliensekről (User Interface, UI) gépekről beküldött feladatokat fogadja, a számítási vezérlők által sugárzott információk alapján nyomon kövesse, hogy mely állomások üzemképesek, kiválasztva ezek közül azt, ahova az adott feladatokat kiküldje, a felhasználót mindig ellássa friss információval a feladatának állapotáról, illetve ha a feladat terminál, a futás eredményét, vagy hibás működés esetén hibaüzenetet eljuttassa hozzá. Ez a három szolgáltatás-típus teljesen általános, de még néhány szolgáltatás szükséges ahhoz, hogy a rendszer teljesen üzemképes legyen.

A továbbiakban használt fogalmak már az EGEE grid megvalósítására vonatkoznak. A BDII szerverek (Berkeley Database Information Index) LDAP (Lightweight Directory Access Protocol) címtárkezelő eljárás segítségével lekérdezhető adatbázisok, melyek a különböző számítási vezérlőkről gyűjtenek információkat és ezt az információt bocsátják az erőforrás elosztók rendelkezésére. Az EGEE rendszerében megjelenik még a MON (Monitor) géptípus, amelynek feladata a megfigyelés és adatgyűjtés a futtatott feladatok számáról, processzor- és memóriaigényéről, futási idejéről stb. Ennek az információnak később az elszámolásnál igen nagy jelentősége lesz, hiszen a virtuális szervezetek kvótái pontosan ezekre az adatokra épülnek.

Az adatorientált grid rendszerek kevésbé változatos struktúrájúak. Egy központi diszk-szerverből (Storage Element, SE) és esetlegesen ehhez csatlakozó diszktárakból áll az állomás, amelyhez vagy állomásonként, vagy régiónként, esetleg centrálisan tartozik néhány katalógus szerver is. Ez lehet fájl-, replika-, vagy teljes logikai katalógus. Feladata minden esetben, hogy nyomon kövesse az óriási mennyiségű, több példányban eltárolt adatokat és könnyű hozzáférést biztosítson ezekhez.

Ha az adatorientált és a számítási gridet kombináljuk, akkor célszerű állomásonként üzemeltetni egy BDII szolgáltatót, ami kapcsolatban áll az úgynevezett központi BDII szerverekkel, így az erőforrás elosztók egységes forrásból tájékozódhatnak mind a számítási kapacitás, mind pedig az adattároló szolgáltatások tekintetében.

2. Az EGEE2 története

Az EGEE2 fennállása óta több nagy kísérlet infrastruktúráját olvasztotta magába. A legfontosabb ezek közül a CERN (Conseil Européen pour la Recherche Nucléaire, Európa Részecskefizikai Kutatólaboratórium) ami jelenleg gigantikus méretű részecskegyorsítóját, az LHC-t (Large Hadron Collider – Nagy Hadron Ütköztető) építi [3]. Ezzel a tudományos kutatás egyik nagy problémáját oldják meg, ugyanis jelenleg a kisebb kutatóintézetek nem bővíthetők tovább, nincs anyagi fedezet egy ekkora tudományos befektetésre, viszont nemzetközi összefogásban mindez megvalósítható. Az LHC üzembe helyezésével Amerikából újra Európába fog átkerülni a részecskefizikai kutatás súlypontja, mivel ez lesz a legnagyobb ütközési energiát elérő részecskegyorsító a világon. A részecskefizika kiemelkedően magas számításigénye miatt a CERN már évek óta fejleszti saját grid rendszerét és komoly tapasztalatra tett szert ezen a téren.

2004 áprilisában – a 2001-től futó EDG (European Data Grid) projekt sikeres zárását követően – a CERN támogatásával aláírásra kerültek az EGEE első két évének fejlesztési célkitűzései és megvalósítási tervei. Az Unió és a CERN kooperációja igen fontos momentuma volt a projekt létrejöttének. A CERN az EDG zárása után az EGEE keretein belül saját grid projektje, az LCG (LHC Computing Grid) fejlesztésével és üzemeltetésével próbálta fedezni a kísérletek jövőbeli adattárolási és számítási igényét, de a közös cél felismerése után az Európai Unió által finanszírozott másik projekttel, a gLite-tal kezdett összeolvadni. Ez a folyamat még ma is tart. Az LCG célja így kissé módosult: jelenleg a nagyenergiás-fizika kutatási területén dolgozóknak kíván infrastruktúrát fenntartani. A CERN így az EGEE fő fejlesztője és több központi szolgáltatás üzemeltetője lett.

Egy másik oka az EGEE megalapításának az európai bioinformatikai kutatások támogatása. A BioMed projektnek három fő kutatási célja van. Az egyik a számítógépes szimulációkon alapuló gyógyszerkutatás. Ennek sikeres eredményei közé sorolhatjuk a malária és a madárinfluenza elleni harc jó néhány friss eredményét. Másik tervezetük az orvosi képalkotás és képfeldolgozás, amelyben egy hatalmas orvosi adatbázist terveznek létrehozni. Részletes, nagy felbontású képeket, kórtörténeteket és egyéb, kezelés és kutatás számára fontos információkat kívánnak összegyűjteni, rendszerezni és ezt a projektben résztvevő orvosok számára – megfelelő személyiségi jogok biztosítása mellett – hozzáférhetővé tenni. Így az orvosok már a páciensek vizsgálata közben láthatják a hasonló tünetű betegek teljes kórtörténetét, még a diagnózis felállítása előtt. A harmadik beruházás a bioinformatika módszereivel történő gén- és proteinelemzés. Ennek a projektnek a részletesebb taglalása viszont meghaladja a cikk kereteit.

1. táblázat Az EGEE résztvevői tevékenységi kör szerinti felosztásban

1. Kommunikációs tevékenységek				
projekt-irányítás (NA1)	információ szétosztás (NA2)	felhasználók képzése (NA3)	alkalmazások és támogatásuk (NA4)	nemzetközi együttműködés (NA5)
2. Szolgáltatási tevékenységek				
grid működtetése, irányítása, erőforrás biztosítása (SA1)			hálózati szolgáltatás (SA2)	
3. Fejlesztési tevékenységek				
szolgáltató programcsomag (Middleware) fejlesztése (JRA1)	minőségbiztosítás (JRA2)	biztonság (JRA3)	hálózatfejlesztés (JRA4)	

3. Az EGEE grid szerveződése

Az EGEE első ülésén lefektették a grid üzemeltetésének alapjait. Több mint 27 ország, közöttük az Egyesült Államok, Japán és Oroszország, valamint összesen 70 közreműködő szervezet csatlakozott egy páneurópai grid kialakításához és üzemeltetésének megszervezéséhez. 35 millió eurós költségvetés került szétosztásra. Ezt nem csak az elinduláshoz szükséges géppark megépítésére szánták, hanem a grid építőkockáit jelentő állomásoknak önálló klaszterként való működéséhez szükséges szoftverek kifejlesztésére, az állomásokat összehangoló, a feladatok kiosztását végző szerverek szolgáltatásainak megírására, az állomásokat felügyelő monitor programok, valamint a felhasználók számára kényelmes elérést biztosító köztes szoftverréteg kialakítására. Egy nagyon bonyolult rendszerről van szó, amely csak komoly hierarchikus szerveződésben, ezer és ezer feladat megvalósításával működhet.

Az EGEE résztvevőit tevékenységi kör szerint három nagy csoportra oszthatjuk, melyeket az *1. táblázatban* vázolunk fel. Magyarországon EGEE grid állomások üzemeltetésével számítástechnikában érintett kutatóintézetek, egyetemek foglalkoznak *(2. táblázat)*.

A jelenleg elérhető erőforrások szerint a magyar grid tevékenység legnagyobb infrastruktúrájával rendelkező KFKI-RMKI 173 CPU-t és 10 Terabájt tárolókapacitást szolgáltat, az ELTE 12 CPU mellett 7 Terabájttal járul hozzá a 200 CPU-nyi és 20 Terabájtnyi teljes magyar kapacitáshoz. A HunGrid VO KFKI RMKI által üzemeltetett honlapja a *http://grid.kfki.hu/hungrid* címen, az LCG-hez kapcsolódó magyar Grid-tevékenység honlapja a *http://www.lcg.kfki.hu* címen található [3].

Ezenkívül az ELTE üzemelteti a HunGrid portált, mely a SZTAKI által kifejlesztett P-GRADE portálon alapul [3]. A HunGrid további számítási és adat kapacitását a BME és a NIIF nyújtja. A SZTAKI és a BME az oktatási tevékenységekben, a konferenciaszervezésekben vállalnak szerepet. A SZTAKI 2005 óta évente szervez egy grid nyári iskolát [3] és 2007-ben megkezdte a GASUC (Grid Application SUpport Centre) grid alkalmazás támogatási szolgáltatását nem csak a hazai, hanem a külföldi alkalmazások támogatására is [3].

Az állomásokat területi szempontból is fel kellett osztani a hatalmas földrajzi kiterjedés miatt, amely azonnal egy természetes hierarchikus irányítási struktúrát is adott a rendszernek. Így a projektekhez csatlakozó intézetek területek, régiók szerint is beosztásba kerülnek. Az egyes régiók önálló működésre is képesek. Így biztosított a 24 órás elérhetőség: ha egy szolgáltatás, vagy szolgáltató kiesik, a rendszer másik része automatikusan átveheti a feladatát. Jelenleg tíz régió van: CERN, Közép-Európa (Csehország, Ausztria, Lengyelország, Magyarország, Szlovákia, Szlovénia), Franciaország, Németország és Svájc, Írország és az Egyesült Királyság, Észak-Európa (Dánia, Belgium, Észtország, Finnország, Hollandia, Norvégia, Svédország), Olaszország, Oroszország, Délkelet-Európa (Görögország, Ciprus, Bulgária, Izrael, Románia) és Délnyugat-Európa (Portugália, Spanyolország).

Létezik még egy felosztás, az adatterjedés és adatáramlási folyamatok szervezése szempontjából. Ez a Tier rendszer. Ennek négy hierarchikusan élesen elkülönülő szintje van. A központok felosztása az erőforrások méretén, a tárolókapacitás mennyiségén alapul. T0 a CERN, az adattárolás központja. A T1 központok a CERN-ből közvetlen, dedikált kapcsolaton keresztül kapják az adatokat. A T2 központok a hozzájuk regionálisan legközelebb eső T1-es központokból jutnak az adatokhoz. A többi állomás a T3-as besorolásba esik, és nem tartozik egyetlen T1-hez sem. A KFKI RMKI jelenleg csatlakozik a CMS kísérlet (Compact Muon Solenoid, az LHC egyik fő kísérlete) T2-es kiszolgálói táborába.

Az EGEE feladata egy egységes grid környezet kialakítása. Az üzemeltetéshez kapcsolódó szoftverek fejlesztését nagy részben a CERN végzi, bár jelentős a más grid-fejlesztők által létrehozott rendszerek EGEE-be importálása is. Ilyen például a Globus Alliance által létrehozott Globus Toolkit, amely az alapvető grid funk-

KFKI RMKI	(KFKI Részecske- és Magfizikai Kutatóintézet)	SA1
NIIFI	(Nemzeti Információs Infrastruktúra Fejlesztési Intézet)	SA1
BME	(Budapesti Műszaki és Gazdaságtudományi Egyetem)	NA2, NA3
ELTE	(Eötvös Loránd Tudományegyetem)	NA2, NA3
MTA SZTAKI	(Számítástechnikai és Automatizálási Kutató Intézet)	NA1, NA2, NA3, NA4, SA1

2. táblázat EGEE grid állomás-üzemeltetők Magyarországon

ciókat szolgáltatja, vagy az EGEE-n belül legelterjedtebb PBS/Torque ütemező, amit a Cluster Resources Inc. fejleszt.

4. Az EGEE grid felhasználói

Az EGEE célkitűzési között szerepel a grid rendszerek felhasználói táborának bővítése. Amit ajánl, az egy magas szintű, biztonságos és dinamikusan fejlődő szolgáltatás. Az EGEE feladata a saját grid rendszerének népszerűsítése és hosszú távon terv az infrastruktúra ipari alkalmazásának megvalósítása is.

A felhasználóknak két feladata van: a regisztrációhoz elengedhetetlenül szükséges egy X.509 típusú felhasználói tanúsítvány. Ez a személyi igazolvány szerepét tölti be grid használata során. A grid rendszert különböző felhasználói csoportok használhatják, ezeket VO-nak (Virtual Organization, virtuális szervezetek) nevezzük. A VO-k egyenként köthetnek megállapodást a állomásokkal, a felhasználni kívánt erőforrások tekintetében. Ilyen értelemben az állomások a szolgáltatók, a VO-k pedig megrendelik a szolgáltatást és a VO felhasználói használják a csoportjuk által megrendelt erőforrást, a szervezeten belül meghatározott jogok és jogosultságok szerint. Ilyen felhasználói csoportot képez például az orvosok és biológusok projektje, a BioMed. A csoportok érdekeltségük szerint különböző mértékben kérnek és kapnak részesedést az erőforrásokból. Például az EGEE keretein belül az LHC CMS kísérletéhez kapcsolódó feladatok prioritása az RMKI állomáson nagyobb, mint egy biogrid alkalmazásé.

Sok szervezet – így például a négy nagy CERN kísérlet is – saját keretrendszert fejlesztett ki felhasználói számára. A magyarországi MTA SZTAKI egyik projektje viszont egy univerzális felhasználói felület kialakítása. Ez a P-Grade portál, mellyel nem csak az EGEE-hez, de jó néhány más grid rendszerhez is lehetősége van a felhasználónak csatlakozni [2]. Ez is jól mutatja, hogy alapvetően a különböző grid rendszerek nem állnak olyan messze egymástól és már létezik egy nemzetközi összefogás egy globális grid szabvány kialakítására.

Az EGEE2 2006-2008 között fut, jelenleg főleg ebből támogatják a hazai EGEE-hez kötődő grid kutatásokat. Az EGEE1-hez képest a felhasználók száma jelentősen nőtt, pillanatnyilag 32 ország 90 intézete vesz részt az infrastruktúra kialakításában és működtetésében. A felhasználható erőforrás is jelentősen gyarapodott: az EGEE2-ben jelenleg több, mint 40000 CPU és mintegy 12 Petabájt tárterület áll a felhasználók rendelkezésére.

A feladatok koordinálása, a különböző nemzetek és üzemeltető csoportok közötti kapcsolattartás levelezési listákon, video- és távkonferenciákon, gyakrabban-ritkábban megtartott közös üléseken történik. A legnagyobb ilyen esemény az éves EGEE konferencia, ahol az üzemeltetők és a felhasználók egyaránt előadják tapasztalataikat, a fejlesztők bemutatják éves munkájukat, illetve a gridet alkotó állomások képviselik magukat. Ennek idén Budapest adott otthont, a BME szervezésében.

5. Magyarország az EGEE2-ben

Magyarország 1992 óta tagja a CERN-nek, illetve 2004 óta az Európai Uniónak is, így sikeres pályázatokkal veszünk részt az EGEE infrastruktúra létrehozásában és üzemeltetésében. Jelenleg több, különböző aktivitásokban részt vállaló intézetünk és néhány közepes és kisebb állomásunk részesül támogatásban az EGEE projektből. Ez azt jelenti, hogy Magyarország jelentős mennyiségű számítási és tárolási kapacitással, valamint fejlesztési és oktatási tevékenységgel járul hozzá a világ jelenleg legtöbb felhasználóval rendelkező grid rendszeréhez.

Az EGEE projekt folytatásaként jelenleg folyó EGEE2 és a meghirdetett EGEE3-ban is szerephez jutnak a magyar résztvevők. Így a magyar grid infrastruktúra további üzemeltetésére, fejlesztésére, a grid fejlesztésekben való részvételre készülünk. Az EGEE projektben való szerepvállalás pedig a magyar kutatók számára is biztosítja az erőforrások nemzetközi hálózatához való hozzáférést.

A hazai projektek közül a HunGrid virtuális szervezet a magyar kutatói szféra érdekeinek képviseletére jött létre. Mivel nem csak EGEE támogatásokból, hanem egyéb pályázati forrásokból is származnak a magyar állomások bevételei és az EGEE célja általános értelemben a kutatás támogatása, meg akartuk teremteni a magyarországi állomások felhasználhatóságát a hazai kutatások számára is. Ennek keretein belül minden magyar akadémiai szférában tevékenykedő kutató, akinek munkájához nagy számítási vagy tárolókapacitásra van szüksége, jogosult az üzemeltető állomásokon hozzáférést és erőforrásokat kérni az EGEE által is támogatott rendszerből.

Irodalom

[1] Ian Foster, Carl Kesselman,
The Grid: Blueprint for a new computing infrastructure, Morgan Kaufmann Publishers, 1998.

[2] Peter Kacsuk, Tamas Kiss, Gergely Sipos:
Solving the Grid Interoperability Problem by P-GRADE Portal at Workflow Level, Proc. of the 15th IEEE Int. Symposium on High Performance Distributed Computing (HPDC-15), Paris, France, 2006.

[3] Hivatalos weboldalak
EGEE: http://www.eu-egee.org/
BioMed: http://www.bioinfogrid.eu/
LCH: http://lhc.web.cern.ch/lhc/
P-GRADE: http://www.lpds.sztaki.hu/pgrade/
Nyári iskola: http://www.egee.hu/grid07/
GASUC: http://www.lpds.sztaki.hu/gasuc/
HunGrid: http://grid.kfki.hu/hungrid
RMKI: http://www.lcg.kfki.hu/

[4] HunGrid „Felhasználói Fórum” előadások:
http://indico.cern.ch/conferenceDisplay.py?confId=15913

SEE-GRID: a dél-európai grid-infrastruktúra

KOZLOVSZKY MIKLÓS, DRÓTOS DÁNIEL, KARÓCZKAI KRISZTIÁN, LOVAS RÓBERT,
MÁRTON ISTVÁN, SCHNAUTIGEL ANDRÁS, BALASKÓ ÁKOS

MTA-SZTAKI, Párhuzamos és Elosztott Rendszerek Kutatólaboratóriuma
m.kozlovszky@sztaki.hu

TÓTH ADRIÁN
Miskolci Egyetem

Lektorált

Kulcsszavak: GRID, SEE-GRID projekt, grid infrastruktúra

Az MTA-SZTAKI Párhuzamos és Elosztott Rendszerek Kutatólaboratóriuma (LPDS) részt vesz a dél-európai grid infrastruktúra kialakításában és üzemeltetésében. A SEE-GRID (South Eastern European GRid-enabled eInfrastructure Development) projekt keretén belül, nemzetközi partnerek közreműködésével már évek óta folyamatosan számítási és tárolási erőforrásokkal bővíti az infrastruktúrát. A dél-európai grid a térség felhasználói és alkalmazásfejlesztői számára szabadon hozzáférhető és elsődleges célja az akadémiai szférában folyó kutatások, valamint oktatási tevékenységek kiszolgálása. Cikkünkben bemutatjuk az egymást követő SEE-GRID projektek során kialakított grid infrastruktúrát, szemléltetjük a kialakított rendszer működését és információkat szolgáltatunk használatának lehetőségeiről, illetve az infrastruktúrán jelenleg folyó nagyobb kutatási projektekről.

1. Bevezetés

A gridrendszerek olyan szolgáltatásokat megvalósító, többnyire heterogén informatikai rendszerek, melyekben a hálózatokkal dinamikusan összekapcsolt egységek (számítógépek illetve egyéb erőforrások), földrajzi helytől függetlenül, egységes módon, jogosultságokkal szabályozva, igény szerint (on-demand) elérhetők a rendszer felhasználói, illetve azok programjai számára. A gridtechnológia lehetővé teszi az elosztott tetszőleges erőforrások egységes kezelését, biztonságos módon történő (újra)felhasználását és sok esetben támogatja a felhasználók kooperatív munkafolyamatait is [1,2]. Jelen cikkben bemutatjuk a dél-európai grid-infrastruktúra kialakításában kulcsszerepet játszó SEE-GRID projektet, szemléltetjük a kialakított rendszer működését, információkat szolgáltatunk használatának lehetőségeiről, illetve az infrastruktúrán jelenleg folyó nagyobb kutatási projektekről.

1.1. Az általános gridrendszer részei

A GRID rendszerek elengedhetetlen eleme a köztesréteg (middleware), mely a működéséhez szükséges belső és külső szolgáltatásokat biztosítja. A grid-infrastruktúrán belül a különböző telephelyek (site-ok) általában az alábbi komponensekkel rendelkeznek:

- *Felhasználói interfész* – UI (User Interface): belépési pont a felhasználók számára.
- *Számolási egység* – CE (Computing Element): feladatvégrehajtási erőforrás, mely többek között a helyi grid-erőforrások ütemezéséért felel, Frontend és Worker gépek.
- *Tároló egység* – SE (Storage Element): tárolókapacitást biztosító szolgáltatást nyújt.

Az alapszolgáltatások mellett esetenként még további szolgáltatásokkal is kiegészítik a telephelyeket, úgymint: Erőforrás bróker (RB, Resource Broker), Információs szolgáltatás (IS, Information Service), BDII (Berkeley DB Information Index), Replika katalógus (RC, Replica Catalog) és Proxy szerver.

1.2. gLite

Az új gLite verzió 3.1 köztesréteg 2007 nyarán került kiadásra (a korábbi Scientific Linux 3-at [3] támogató gLite 3.0 2006 májusában adták ki) és mindamellett hogy ez a verzió már Scientific Linux 4-et támogat, sok kedvező tulajdonságot örökölt a korábbiakban kifejlesztett EDG és az LCG köztesrétegekből. Főbb tulajdonságai közül lényeges kiemelni hogy nyílt forráskódú szoftver, kompatibilis számos egyéb technológiájú ütemezővel (pl. Condor, PBS), moduláris felépítése követi a SOA (Service Oriented Architecture) elveket, valamint korábbiaktól eltérően az alacsony szintű gridszolgáltatások mellett már támogat magasabb szintű (pl. DAG típusú munkafolyamat-futtatási) szolgáltatásokat is. Főbb szolgáltatás részei az:

– információs és nyomkövető szolgáltatások,
– adatkezelő szolgáltatások,
– biztonsági szolgáltatások,
– kiegészítő és feladatkezelő szolgáltatások,

melyek elvégzik az adatmenedzsment, terhelésmenedzsment, információmenedzsment, felügyelet, számlázás, naplózás, könyvelés, hálózat-felügyelet, adatgyűjtés feladatköreit.

1.3. Történelem – a SEE-GRID projekt

A SEE-GRID projekt 2004-ben indult több mint 1,2 millió eurós EU által támogatott költségvetéssel, 11 ország részvételével (Albánia, Bosznia-Herzegovina, Bulgária, Horvátország, Macedónia, Görögország, Magyarország, Románia, Szerbia-Montenegró és Törökország). A projekt első fázisában a partnerek szoros együttműködésben kiépítették az alapinfrastruktúrát. A kitűzött kezdeti célok között az alábbiak szerepeltek:

- A dél-európai grid-infrastruktúra felépítése, az infrastruktúra üzemeltetéséhez szükséges támogató rendszerek kialakítása.
- A személyes kapcsolati hálózat kialakítása az egyes projekt országok gridkutatói között.
- A nemzeti grid-infrastruktúrák (NGI) kiépítésének támogatása.
- A digitális megosztottság mérséklése a dél-európai és nyugat-európai régiók között.
- A grid terjedésének és használatának elősegítése a dél-európai országokban, együttműködés más grid-projektekkel, kiemelten az EGEE-vel (Enabling GRIDs for E-sciencE), a legnagyobb európai grid-projekttel.

1.4. A SEE-GRID2 projekt

Időközben történelmi okokból, valamint újabb tagok felvétele miatt a projekt-konzorcium partnerségi viszonyai átalakultak [4].

A 2006-ban elindított SEE-GRID2 projekt a sikeres SEE-GRID projekt folytatásaként 2 millió eurós megnövelt projekt-költségvetéssel már 13 tagot (új tagok: Montenegró és Moldova) foglalt magában, melyekhez időközben 27 külső egyetem, illetve kutatólaboratórium kapcsolódott. Fő célként tűzte ki a 24/7 típusú grid-szolgáltatás biztosítását a dél-európai térség akadémiai kutató intézményeinek, valamint a gridet alkalmazó felhasználói- és alkalmazás-fejlesztői közösség létszámának tovább növelését. A SEE-GRID-2 projekt 2008 második negyedévében fog lezárulni.

2. SEE-GRID Infrastruktúra

A SEE-GRID infrastruktúra 31 teljes értékű, valamint 4 hitelesítés alatt álló (Albánia 1, Horvátország 1, Románia 2) csomópontjaival a legerősebb dél-európai grid infrastruktúraként lefedi az összes tagország területét *(1. ábra)*. A SEE-GRID infrastruktúra jelenleg (2007. november) gLite 3.0.2 köztesréteget használ. A processzorok száma meghaladja a 950-et, dedikált háttérkapacitásai pedig elérik a 24 Terabájtot *(2. ábra)*.

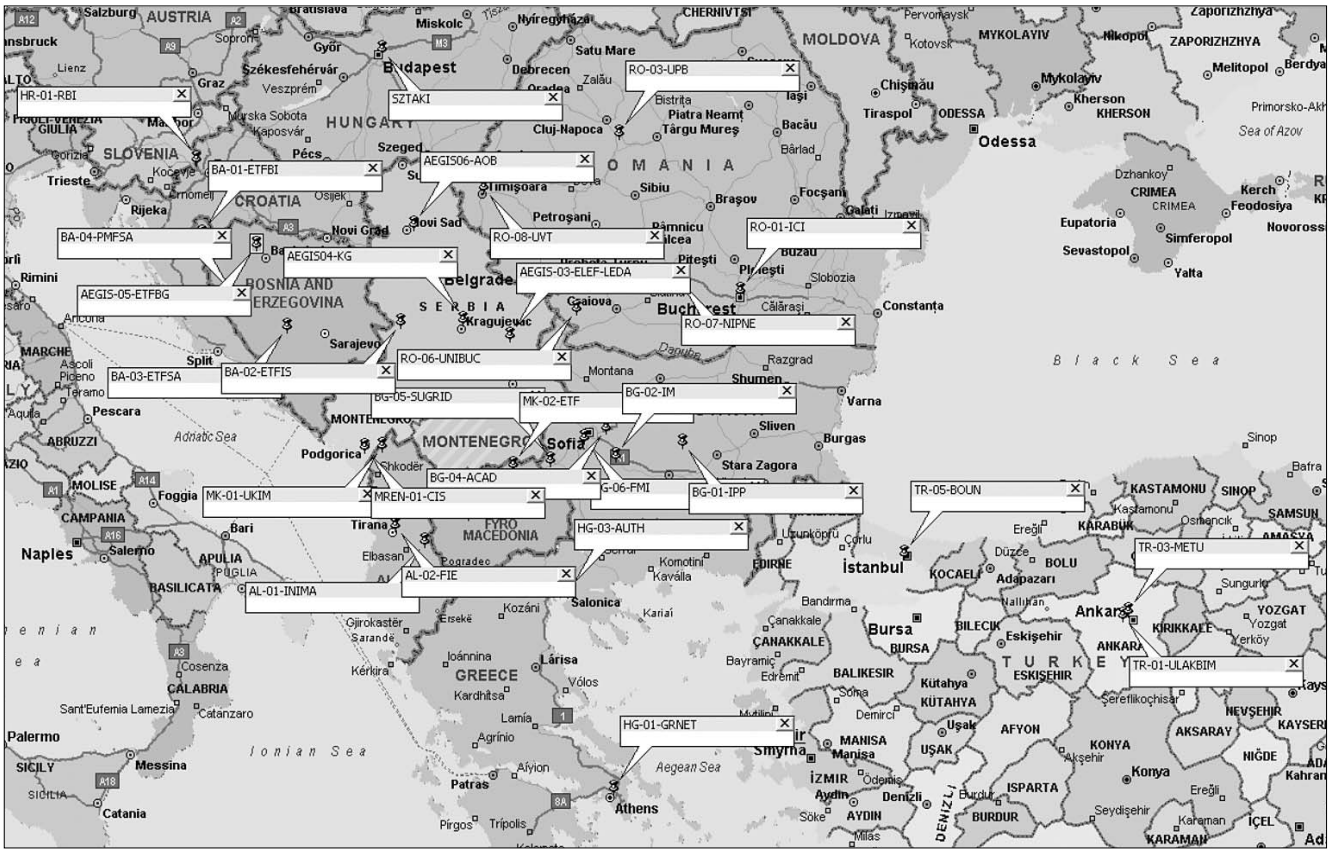
2.1. Hálózati réteg

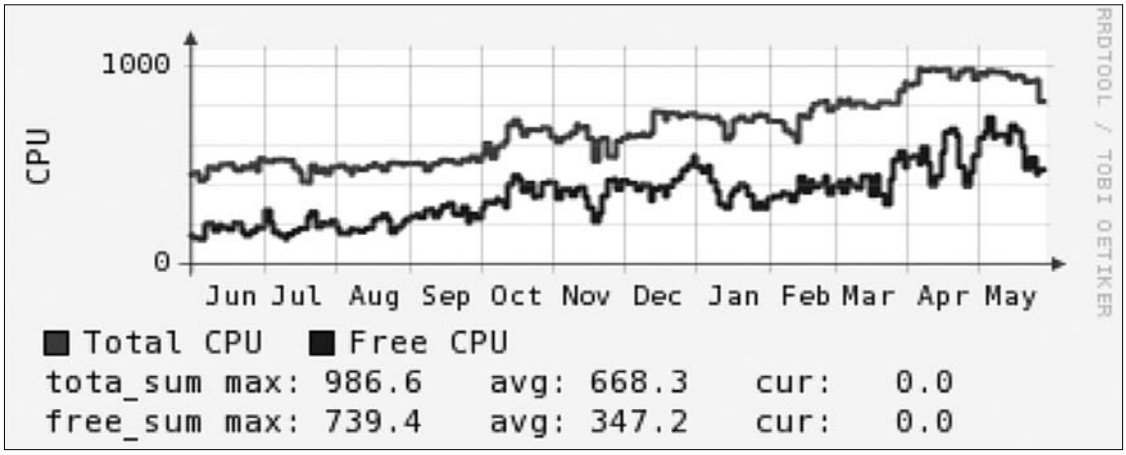
A SEE-GRID projekt grid infrastruktúra csomópontjai között folyó kommunikáció is túlnyomórészt a GEANT(2) által kiépített hálózaton keresztül valósul meg. Az EU által támogatott GÉANT-2 [5] (a pán-európai hálózat 7. generációja) olyan 34 európai országra kiterjedő hibrid, több gigabit sebességű hálózat, mely lehetővé teszi a hálózaton belüli kutató és oktató központok nagy sebességű közvetlen összekapcsolását, valamint Észak-Amerika, Japán, Dél-Amerika, a mediterrán régió, a Közel-Kelet, Dél-Afrika és az ázsiai régió jelentős részének kutatói közösségei felé is nagykapacitású hálózati összeköttetést biztosít. A hálózati infrastruktúra a DANTE (Delivery of Advanced Network Technology to Europe) [6] üzemeltetése alatt áll.

2.2. Hozzáférés-engedélyezés

Grid-infrastruktúrákban elterjedten használnak mind a felhasználók, mind pedig feladataik azonosításához

1. ábra A SEE-GRID infrastruktúra





2. ábra SEE-GRID processzorok számának változása az idő függvényében (2006. június-2007. május)

tanúsítványokat. Az akadémiai/tudományos gridekben az X509 alapú tanúsítvány a leggyakrabban alkalmazott, melyek kiadását és menedzselését úgynevezett tanúsítványhatóságok/hitelesítő központok (CA, Certificate Authorities) végzik. A SEE-GRID2 project keretén belül összesen 15 CA teljesít szolgálatot, a központit a görög projektpartner működteti. Magyarország a SEE-GRID infrastruktúrában elfogadott saját önálló CA-val rendelkezik, melyet a NIIFI Tanúsítvány Hitelesítő Szolgáltatásának részeként üzemeltet az alábbi honlapon: *http://www.ca.niif.hu*.

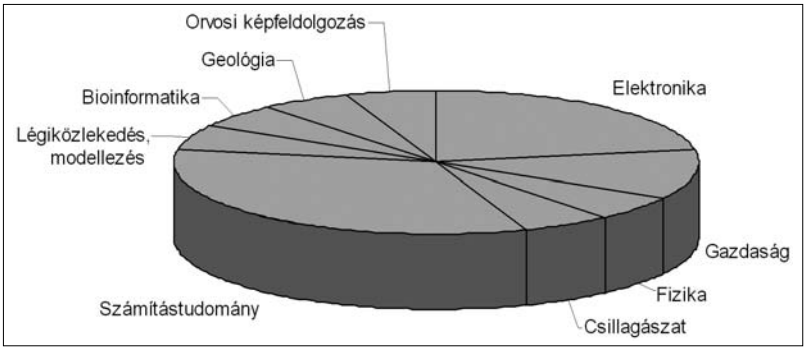
2.3. Az infrastruktúra tesztelése és monitorozása

A projekt-partnerek által közös megegyezéssel előre definiált szolgáltatási színvonal (SLA) alapján történik a grid-infrastruktúra működtetése. Az elosztott felügyeleti rendszer a problémák kapcsán beküldött hibajelentések (úgynevezett ticketek) alapján dolgozik. A projekt-partnerek egy hetes intervallumokban rotációs rendszerben manuálisan is tesztelik/ellenőrzik az infrastruktúrát. A grid-infrastruktúra működésének monitorozására, valamint az egyes gridhelyek funkcionális ellenőrzésére különféle tesztelési metódusok, illetve tesztelő rendszerek használatával kerül sor, melyek nagy része a projekt saját fejlesztésű teszteszköze *(3. ábra)*.

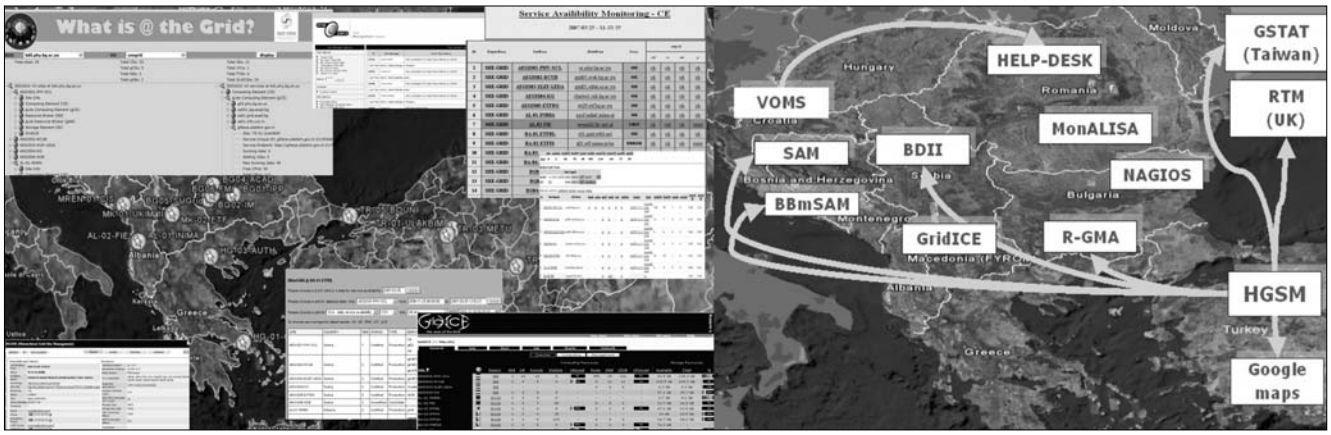
3. SEE-GRID-2 alkalmazások

A SEE-GRID-2 projekt széles körű támogatást biztosít mind a gridalkalmazások, mind pedig a grid-alkalmazások fejlesztők számára. A projekt partnerek által kiemelt fontosságúként megjelölt gridalkalmazások számára dedikált, úgynevezett e-infrastruktúrát biztosít, fejlesztésükhöz nemzetközi szakértő gárdájával (ASG, Application Support Group) támogatást nyújt, és garanciával vállalja az alkalmazások szolgáltatásszerű futtatását az infrastruktúrán. Jelenleg 18 kiemelt alkalmazás *(4. ábra)* ellenőrzött futtatását, valamint továbbfejlesztését támogatja a SEE-GRID-2 projekt, melyekhez a későbbiekben a projekt-partnerek igényeinek megfelelően újabb alkalmazások fognak csatlakozni.

4. ábra
Kiemelten támogatott alkalmazások szakterületei



3. ábra A SEE-GRID infrastruktúra működés monitorozási és funkcionalitásainak tesztelési környezete [7]



3.1. Magyarországi SEE-GRID-2 alkalmazások

Több magyar felsőoktatási intézmény is aktívan részt vesz a SEE-GRID-2 projektben alkalmazások gridesítésével, illetve grides alkalmazásfejlesztéssel.

- A Miskolci Egyetem kutatói a FEM2.5D (Dimensional Frequency Domain Electromagnetic Numerical Modelling) [8] kódnevű alkalmazás gridesítésén dolgoznak, mely elektromágneses terek (2D/3D) modellezését végzi.
- A Nemzeti Üzleti Főiskola kutatói az EMMIL (E-Marketplace Model Integrated with Logistics) [9,10] kódnevű B2B típusú alkalmazás gridesítését végezik a projekt során, melynek segítségével internetes aukciók komplex, nagy paraméterterű logisztikai problémáira lehet optimális megoldásokat keresni algoritmikusan.

4. Hozzáférés az infrastruktúrához

Az alábbiakban összefoglaljuk azokat a lépéseket, melyeket a felhasználóknak, illetve fejlesztőknek meg kell tenniük a SEE-GRID erőforrások használatához [11]:

1) Magyarországi felhasználók esetén a SEE-GRID infrastruktúra használatához szükséges tanúsítványt a NIIFI-től kell igényelni az intézet honlapján [12].

2) A felhasználók tanúsítványuk segítségével a jogosultságuknak megfelelő virtuális szervezet (jelen esetben a SEEGRID VO) infrastruktúrájához a VO tagság igénylése után hozzáférhetnek és ezen erőforrásokat felhasználhatják saját alkalmazásaik futtatásához. VO tagság igénylését a korábbiakban már megszerzett tanúsítvánnyal a *https://voms.irb.hr:8443/voms/seegrid/webui/request/user/create* honlapon lehet megtenni.

3/a) A VO tagság és az érvényes tanúsítvány megszerzését követően a SEE-GRID infrastruktúra szolgáltatásai a P-GRADE Portál alapú SEE-GRID Portálon *(5. ábra)* keresztül érhetők el az alábbi helyen: *http://portal.p-grade.hu/seegrid*.

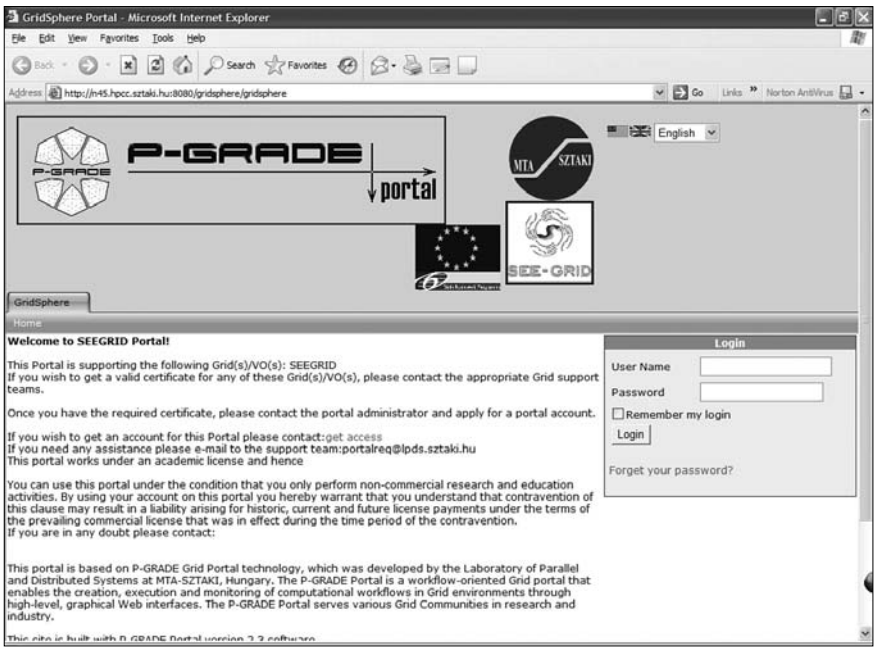
Ez a Magyarországon kifejlesztett és üzemeltetett portál képes kiszolgálni a teljes dél-európai felhasználói kört. Érdemes itt kiemelni, hogy Magyarország nemzetközi mércével mérve is igen erős a grid portálok fejlesztésében, melynek eredményeképpen több, egymással párhuzamosan kifejlesztett gridportál is szolgálja a nemzetközi felhasználói csoportokat. A legnagyobb felhasználói táborokon a BME által fejlesztett (jelen lapszámban is bemutatásra kerülő), GridSphere Portál Keretrendszerre épülő Conflet rendszer és az MTA SZTAKI LPDS által fejlesztett P-GRADE gridportál megoldások osztoznak.

A P-GRADE gridportál az utóbbi másfél évben nemzetközi méretű projektté nőtte ki magát; egyes moduljai horvát, angol, illetve török fejlesztéssel készülnek és világszerte (többek között Svájc, USA, Anglia, Olaszország) 14 működő portál szolgálja ki a gridfelhasználókat. A SEE-GRID Portál (P-GRADE portál alapú) felhasználói felülete bármilyen hagyományos Web-böngészővel használható a felhasználó földrajzi helyétől, illetve a kliens operációs rendszer típusától függetlenül. A SEE-GRID Portál segít az erőforrások kiválasztásában, az erőforrások terheltségének vizsgálatában, a munkafeladatok gridbe történő elküldésében és felügyeletében.

Azonosító igénylés – formanyomtatvány kitöltésével – a *http://portal.p-grade.hu/index.php?m=9&s=1* helyen lehetséges. Bővebb információkat a portálról, illetve annak kezeléséről a *portalreq@lpds.sztaki.hu* címre küldött email-el lehet kérni.

3/b) A VO tagság és az érvényes tanúsítvány megszerzését követően a parancssoron keresztül történő grid használat egy megfelelően beállított UI gépen keresztül (mely legtöbbször az adott országbeli projektpartnernél üzemel, és melyhez legtöbbször azonosító igénylése szükséges) történhet.

5. ábra A SEE-GRID portál üdvözlőképernyője



5. Alkalmazásfejlesztés a SEE-GRID infrastruktúrán

5.1. Alkalmazásfejlesztés önállóan

A grid-alkalmazások fejlesztői számára levelező listák, weben elhelyezett elektronikus dokumentációk (wiki oldalak [13] és felhasználási útmutatók), információs portálok [14] állnak rendelkezésre, melyek hasznos segítséget nyújthatnak a fejlesztés minden stádiumában.

A SEE-GRID-2 projekt folyamatosan konferenciákat szervez, konferenciákon vesz részt, valamint nemzetközi oktató csapatával előadássorozatokat és tanfolyamokat tart a dél-európai régió országaiban. A

partnerországok területén tartott tanfolyamok a gridfelhasználók, a grid-alkalmazások fejlesztői és a grid-adminisztrátorok három nagy célcsoportjára fókuszálnak.

A SEE-GRID projekthez kapcsolódó konferenciákról a SEE-GRID projekt honlapján található „Events" menüsorból (*www.see-grid.eu/events.php?language=en*) lehet értesülni. Az elmúlt másfél évben a grides technológiáról szóló oktatási események száma elérte a 60-at. Az aktuális grid-oktatási eseményekről a SEE-GRID projekt hivatalos grid-tanfolyamokkal foglalkozó honlapján lehet részletes információkat szerezni:

http://www.lpds.sztaki.hu/seegridtrainingcenter

5.2. Alkalmazásfejlesztés támogatással

5.2.1. GASUC

A SZTAKI Párhuzamos és Elosztott Rendszerek Laboratóriuma által 2007. elején megalapított Grid Alkalmazás Támogató Központ (GASUC, Grid Application SUpport Centre) elsődleges célja meglévő alkalmazások „gridesítésének" támogatása.

A központ munkája során kiemelt hangsúlyt kap a SZTAKI-ban felhalmozott grid-alkalmazásfejlesztési tapasztalatok átadása, a grid-alkalmazók és -alkalmazásfejlesztők táborának szélesítése. A támogató központ gridszakértői nyolc szabványosított lépésből álló munkafolyamat során segítenek az alkalmazás első gridesített prototípusának kialakításában. Bővebb információ a gridifikációs munkafolyamatokról, illetve a központ működéséről a következő oldalon található:

http://www.lpds.sztaki.hu/gasuc/.

5.2.2. SEE-GRID-2 alkalmazástámogató csoport

A SEE-GRID-2 projekt támogatása kiterjed alkalmazások „gridesítésére", illetve már korábbiakban „gridesített" alkalmazások működésének optimalizálására. A projekt partnerek által kiemelt fontosságúként megjelölt grid alkalmazások fejlesztéséhez nemzetközi szakértő gárdájával (ASG, Application Support Group) biztosít támogatást.

6. Összefoglalás

Cikkünkben részletesen bemutattuk a dél-európai SEE-GRID-2 projektet, a projekt által megvalósított grid-infrastruktúrát, az infrastruktúra monitorozását végző szoftvereket, az infrastruktúrán jelenleg futó alkalmazásokat, valamint a SEE-GRID infrastruktúra eléréséhez és használatához szükséges főbb lépéseket.

A SEE-GRID-2 projekt hivatalosan 2008 májusáig működik, a felépített infrastruktúrát a továbbiakban a SEE-GRID-SCI projekt veszi át. A SEE-GRID-SCI projekt 2008-2010 között elsődleges feladatának tekinti a régió grides alkalmazásfejlesztéseinek támogatását, kiemelt tekintettel a meteorológia, szeizmológia és környezeti modellezés (földmágnesesség, klímamodellezés) kutatási területekre.

A SEE-GRID-SCI projekt magyarországi partnereként az MTA SZTAKI Párhuzamos és Elosztott Rendszerek Laboratóriuma fogja koordinálni az ELTE Meteorológiai Tanszékével és az MTA Geodéziai és Geofizikai Kutató Intézetével kialakított közös kutatómunkát, valamint folyamatos támogatást biztosít majd regionális szinten is a felmerülő alkalmazások gridesítéséhez.

Köszönetnyilvánítás

Az alkotók ezúton mondanak köszönetet a SEE-GRID-2 projektnek, a cikkben ismertetett munkák támogatásáért. A SEE-GRID-2 projektet az Európai Bizottság a „Kutatási infrastruktúrák" 031775 szerződésszámú FP6-os projektjének keretében finanszírozza.

Irodalom

[1] Dr. Szeberényi I.:
A Grid technológia és kutatás hazai és nemzetközi eredményei, IX. Országos Neumann Kongresszus, Győr, 2006.
[2] Kacsuk P.:
A magyar grid rendszerek és fejlesztési irányaik, MTA SZTAKI, 2006.
[3] http://de.wikipedia.org/wiki/Scientific_Linux
[4] D. Kotsokali:
SEEGRID2-D5.2-a-2006-07-31 – Promotional package, July 2007.
[5] http://www.geant2.net/
[6] http://www.dante.net/
[7] A. Balaz:
WP3 – eInfrastructure expansion and operations support, SEE-GRID2 1st Project Review, Brussels, June 2007.
[8] Pethő G., Ficsór L., Szabó I.:
Comparison of 2-D VLF and 2.5-D HED's far field regime EM fields, microCAD 2006 International Scientific Conference, Section B: Geoinformatics Spatial Informatics&Mineral Resources, Miskolc, Hungary, pp.35–40.
[9] Dr. L. Kacsukné Bruckner, G. Hermann:
On the Algorithms of the Grid-Based EMMIL E-Marketplace Model, Mipro 2005.
[10] L. Kacsukné Bruckner, T. Kiss:
„Using Grid-technology to Implement an e-Marketplace Integrated with Logistics", Dapsys International Conference, Budapest, 2004.
[11] M. Kozlovszky:
GRID felhasználó és alkalmazásfejlesztő tréning, Budapest, 2007. április 4.
http://shinobu.lpds.sztaki.hu/indico/conferenceDisplay.py?confId=22
[12] http://www.niif.hu/hu
[13] http://wiki.egee-see.org/index.php/SEE-GRID_Wiki
[14] http://www.see-grid.eu/

A grid hálózatok biztonsági kérdései

KŐVÁRI KÁLMÁN

KFKI Részecske- és Magfizikai Kutatóintézet
dalion@sunserv.kfki.hu

Lektorált

Kulcsszavak: adatbiztonság, betörés, incidens, sebezhetőség, X.509, grid, proxy, klaszter

Számítógépes klasztereket egységes nemzetközi hálózatba kapcsolva, megfelelő bróker- és menedzserrétegek közbeiktatásával látszólag egyszerűen létre lehet hozni egy grid rendszert. Viszont a biztonsági szempontból könnyen védhető, zárt és gyakran a világhálóról is leválasztott helyi klaszterekkel szemben a sokszor publikus internet kapcsolatokat használó grid rendszerek komoly biztonsági kérdéseket vetnek fel. A cikk erre a kérdéskörre igyekszik rávilágítani.

1. Bevezetés

„A biztonság folyamat, nem pedig termék." Bruce Schneier ezen mondatát rengetegen idézték már [1], hiszen rámutat a fogalom körül keringő legnagyobb tévhitre. Egy terméket meg lehet vásárolni, ezután lehet használni vagy nem használni és amennyiben már elavultnak érezzük, és újabb verziót látunk megjelenni, frissíthetjük, vagy maradhatunk a régi és jól megszokott változatnál. A biztonsági kérdések tekintetében efféle viselkedést sajnos nem engedhetünk meg magunknak. A biztonság egy folyamat, tehát állandó aktív figyelmet igényel, folyamatosan és éberen kell kezelnünk az érintett eszközöket és szolgáltatásokat. A tapasztalat azt mutatja, hogy az a biztonsági szakember, aki még soha nem volt érintett incidensben, vagy szerencsés, vagy nagyon fiatal. A jó szakember ismérve, hogy minden esetben törekszik a problémák és incidensek megelőzésre, de képes ezek elhárítására és utólagosan a megfelelő reakcióra is. Az utóbbi kettő legalább annyira fontos, mint maga a megelőzés, hiszen ezzel kerülhető el az esetlegesen még súlyosabb másodrendű károkozás és a probléma esetleges megismétlődése.

A biztonság szót sokszor, sok esetben teljesen különböző jelentéstartalommal használjuk, viszont minden esetben valamilyen értelemben a bizalomhoz kapcsolódik. Lehet szó például adatbiztonságról, ezen belül sem mindegy, hogy az a fontos, hogy az adat megmaradjon akár atomtámadás esetén is, vagy az, hogy ténylegesen korlátozott és szabályozott legyen a hozzáférhetősége. Lehet szó biztonságos azonosításról, ahol a fő feladat egy kliens minél megbízhatóbb azonosítása, bizonyos feladatokra való autorizálása, illetve egy személy által delegált másik személy vagy szolgáltatás azonosítása.

A kapcsolatok biztonságának megítélése a két végpont közti hálózati összeköttetés megbízhatóságát, kódoltságát és kívülállók számára való hozzáférhetőségét jelenti. A szolgáltatási biztonság a kliens oldalon felmerülő kérdés: mennyire bízhat meg a felhasználó abban, hogy az általa kért művelet ténylegesen végrehajtódik a rendszerben.

A biztonságot csökkentő tényezők lehetnek fizikai jellegűek, például egy hálózati kapcsolat korlátozott sebessége, de többnyire inkább logikai eredetűek, és vagy vezérlési, vagy biztonsági megfontolásból vannak jelen. A felhasználónak általában csak két apró igénye szokott lenni: mindent és azonnal. Ezzel pedig mindig szemben áll a helyi biztonsági szakértő, aki rögtön azt mondja, hogy semmit és soha, mert úgy biztonságos. Természetesen egyik megközelítés sem helyes, mindig valamiféle kompromisszum a megoldás. Ezen kérdések nem csak a gridek világában aktuálisak, de ezek nagy komplexitása miatt itt hangsúlyozottan jelennek meg. A jelen cikkben ezeket a kérdéseket tekintjük át.

2. Azonosítás, jogosultsági kérdések

A grid világában is szükséges a személyazonosságunk igazolása. Ennek megvalósítására a legtöbb grid-implementáció az X.509 típusú digitális tanúsítványok által biztosított PKI-t (Public Key Infrastructure) választotta. Röviden annyit kell tudni ezekről a tanúsítványokról, hogy egy titkos és egy nyilvános kulcsból állnak, a titkos kulcsot a felhasználó saját feladata titokban tartani, ezért többnyire jelszóval vagy PIN kóddal is védik. A titkos kulcs kompromittálódása vagy annak gyanúja esetén a tanúsítványt azonnali hatállyal vissza kell vonni. A nyilvános kulcsot egy CA (Certification Authority – tanúsítványhitelesítő) a felhasználó azonosítása után hitelesíti, azaz a saját titkos kulcsával aláírja. Ennek a digitális aláírásnak a hitelessége a CA nyilvános kulcsának birtokában ellenőrizhető. A felhasználónak így csak néhány CA nyilvános kulcsának valódiságában kell bíznia, ezek többnyire az operációs rendszerrel együtt, annak részeként letölthetőek. A módszerrel nem csak személyek, hanem szolgáltatások vagy számítógépek kilétének megbízható ellenőrzése is megoldható.

A grid-felhasználók virtuális szervezetekbe (VO, Virtual Organization) tömörülnek. A virtuális szervezetek taglistája nyilvános információ, a tagok DN-jével (Distin-

guished Name – a személyi szám digitális megfelelője) együtt. Minden erőforrás-szolgáltató maga döntheti el, hogy mely virtuális szervezeteket támogatja. Ezekből csak néhány darab van, így ezek felsorolásával könnyen szabályozható a hozzáférés. Alapvetően egy támogatott virtuális szervezet minden tagja jogosult a kérdéses erőforrások valamilyen mértékű felhasználására, viszont minden szolgáltatónak lehetősége van akár személyre szabottan is módosítani az erőforrásaira vonatkozó jogosultságokat.

A pontos erőforrás-hozzáférési beállítások már erősen függnek az alkalmazott grid-rendszertől, de általánosan jellemző, hogy lehetőség van virtuális szervezetek számára dedikálni erőforrásokat – így csak ők használhatják, – illetve úgynevezett *fair share* elven megosztani, ami azt jelenti, hogy bizonyos idő-intervallumonkénti átlagban adható meg, hogy melyik VO milyen mértékben veheti igénybe az adott szolgáltatást.

3. Adatbiztonság

A feladat egyszerű: a felhasználó azt szeretné, hogy az adatai olyan helyen legyenek, ahol nem veszhetnek el egy esetleges diszkhiba, áramkimaradás, vagy rosszindulatú cselekedet hatására. Erre elég egyszerű megoldás, ha az adatot több példányban tároljuk, azaz replikáljuk. A módszer nehézsége egyrészt abban rejlik, hogy a másolatokat folyamatosan frissíteni kell, ha az eredeti adaton változtatunk, másrészt az egyes példányokat az egységes elnevezés mellett is meg kell tudnunk különböztetni egymástól. A replikáció előnye viszont, hogy a skálázott elérhetőséget is megoldja: minél több replika áll rendelkezésre egy adathalmazról különböző földrajzi illetve hálózati pontokon, annál többen férhetnek hozzá egyszerre az adatokhoz. Ezeket a szolgáltatásokat valósítják meg általában az úgynevezett Data Grid rendszerek.

A nehezebb feladat a hozzáférhetőség problémája. Ezt több szinten lehet megoldani. Ha elég a klienseknek az, hogy ők maguk meghatározhatják, hogy a rendszer kinek adjon engedélyt az adat olvasására, akkor ezt egy viszonylag egyszerű ACL (Access Control List – jogosultság lista) alapú hozzáférési sémával meg lehet oldani. Viszont egyes kliensek – kiemelhető példaként a biomedika, konkrétan az orvosi adatok – megkövetelik azt, hogy maga a rendszergazda se férhessen hozzá az adathoz az ő engedélyük nélkül.

Ilyen feladatokat oldanak meg a Hydra tárolórendszerek. Működésük lényege, hogy az adat kódolt formában kerül feltöltésre és replikálásra. A dekódoló kulcsot három részre osztják és a darabokat három különböző kulcs-szerveren helyezik el, szigorú szabályozással, illetve további kódolással biztosítva, hogy csak az arra jogosultak férhessenek hozzá. Ezt a korábban említett PKI használatával lehet megoldani oly módon, hogy az összes hozzáférésre jogosult személy digitális tanúsítványának nyilvános kulcsával titkosítjuk a dekódoló kulcs megfelelő darabját. Ezek után csak a kérdéses tanúsítványok titkos kulcsának birtokában lehet hozzáférni a dekódoló kulcs egy-egy darabjához. Mivel a három kulcs-szerver mindegyike, egyenletesen elosztva 2-2 kulcs-darab birtokában van, így egyetlen kulcsszerver kompromittálódása nem jelenti a kulcs kompromittálódását, másrészt egyetlen szerver elérhetetlensége – esetleg megsemmisülése – sem vezet az adat elvesztéséhez. Természetesen a kulcs-tördelést ennél több szerverre is el lehet végezni úgy, hogy a tördelésnek ezen tulajdonsága megmaradjon, de akár több szerver kiesését is tolerálja a rendszer.

Meg kell azonban jegyeznünk, hogy a módszernek két jelentős hátránya is van: az egyik, hogy a dekódoló kulcsot mindenkinek ki kell adni, aki az adatot olvasni akarja, s ha akár egyetlen kulcs is kompromittálódik, veszélybe kerül az adat. A másik probléma, hogy nem lehetséges az olvasási jog visszavonása, ha egy kliens egyszer már megszerezte a kulcsot. Ezekre a nehézségekre születtek már megoldási ötletek, ilyen például az, hogy a dekódoláshoz szükséges szimmetrikus kulcsot az adatot tároló szerver kapja meg közvetlenül, a felhasználók pedig ideiglenes, egyedi kulcsokat használnak a hozzáféréshez. Ebben az esetben a tároló szervernek is nagyon megbízhatónak kell lennie és mivel minden ilyen biztonsági szintű adatfolyamhoz két kódolást is el kell végezni az adott gépen, nagyon nagy számítási kapacitásra is képesnek kell lennie. Ennek a módszernek a megvalósítása és tesztelése még folyamatban van.

4. Proxy-k, delegáció

Amikor a griden dolgozunk, elemi számítási egységeket, úgynevezett feladatokat (job) küldünk be. A feladatok a küldés pillanatától fogva a terminálásig többnyire 4-5 számítógépet is érintenek. Ezek közül többnek meg kell győződnie a feladat tulajdonosának személyazonosságáról. Ezt nem közvetlenül a digitális tanúsítvánnyal oldjuk meg, hanem egy köztes hírvivő entitást delegálunk, így nem kell kiadnunk a kezünkből a tanúsítványunk titkos kulcsát.

Természetesen egy nem-elsőgenerációs grid szolgáltató programcsomag (middleware) az alábbi lépéseket már automatikusan megteszi helyettünk, de fontos, hogy értsük a háttér-mechanizmust, hogy rálátást nyerjünk a felmerülő biztonságtechnikai kérdésekre. A titkos kulcsot nem adhatjuk ki a kezünkből, a tanúsítványról márpedig csak annak segítségével lehet eldönteni, hogy tényleg hozzánk tartozik-e.

A megoldás ilyen esetekben az, hogy egy új kulcspárt generálunk és ennek segítségével létrehozunk egy rövid – tipikusan 12-24 óra – érvényességű helyettesítő tanúsítványt, úgynevezett *proxy*-t. A proxy nyilvános kulcsát aláírjuk a saját titkos kulcsunkkal, ez a CA aláírásához hasonlóan ellenőrizhető, viszont a titkos kulcsot nem lehet belőle rekonstruálni, így ha mellékeljük hozzá a mi nyilvános kulcsunkat, amelyet a CA aláírt és a szolgáltató gép is bízik az adott CA-ban, akkor a CA nyilvános kulcsával ellenőrizheti a mi tanúsítványunk hitelességét és a mi tanúsítványunk nyilvános kulcsával a proxy hitelességét. A proxy titkos kulcsát jelszavas vé-

delem nélkül, de a lehetőségekhez mérten – például az operációs rendszer által biztosított kizárólagos olvasási joggal – védetten lehet tárolni, így a proxy létrejötte után jelszó gépelése nélkül tudjuk magunkat azonosítani, illetve az általunk létrehozott feladat tudja saját magát azonosítani az ezt igénylő szolgáltatások felé.

Felmerül a kérdés: mi értelme ennek a köztes lépésnek? Maga a proxy kompromisszum a biztonság és a használhatóság között: ésszerű kockázat mellett mentesít a jelszó sokszori begépelése alól. A proxy érvényességi idejének rövidre vételével csökkenthető a vállalt kockázat, hiszen a proxy birtokosa teljes mértékben gyakorolhatja a proxy kibocsátójának jogait; másrészt viszont, ha a proxy lejár, mielőtt a feladat lefutna, az ütemező törli a munkánkat, mivel érvényes proxy nélkül nem vagyunk jogosultak a rendszer használatára.

Ennek a kockázatnak a mértékét tovább lehet csökkenteni speciális proxy tanúsítványokkal, amelyek csak bizonyos állomásokra vagy bizonyos feladatokra érvényesek, de ezek használatával az általános felhasználhatóságot is jelentősen megszorítja az adott grid-rendszer. Ebben a témakörben az aktuális fejlesztések a megszorítások és a biztonság elfogadható kompromisszumát keresik.

5. Szolgáltatási biztonság

Attól a ponttól, hogy beküldtünk egy munkát egy grid-rendszerbe, nincs módunk a további sorsának irányítására. Egyetlen módosítási lehetőség van csak: a feladat törlése a rendszerből (noha bizonyos gridek lehetővé teszik egy feladat szüneteltetését is, ez nem változtat a későbbi lefutásán). Az elvárás tehát az, hogy ha egy feladat bekerült a rendszerbe, akkor fusson is le.

Mivel a rendszer jóval összetettebb, mint egy helyi klaszter vagy szuperszámítógép, egy-egy hiba esetenként előfordulhat. Ezen ritka hibák kiszűrésére a „production” színvonalú grid-rendszerek egytől egyig megvalósítottak valamilyen monitorozó rendszert, amely akár a különböző szolgáltatások logjait elemezve, akár bizonyos időközönként beküldött teszt feladatokkal, vagy a komponensek működését külön-külön ellenőrző funkcionális tesztekkel vizsgálja, hogy mennyire megbízható az infrastruktúra.

Érdekességként megemlítendő, hogy az esetlegesen észlelt hiba sokszor nem a grid-infrastruktúra rendellenességét mutatja, hanem gyakran a hálózat ideiglenes zavarából vagy épp a tesztelő program helytelen működéséből származik.

6. Összefoglalás

Összességében elmondhatjuk, hogy a grid-rendszerekben, mivel természetüknél fogva igen komplexek, mindig könnyen bukkanhat fel hiba, amely biztonsági kockázatot jelenthet még akkor is, ha a felhasználók és a kliensgépek tökéletes viselkedését feltételezzük. Ezért hangsúlyozzuk, hogy minden grid-implementációban nagyon komoly figyelmet kell fordítani a biztonsági kérdésekre, hogy elkerülhetőek legyenek az esetenként igen súlyos incidensek.

Irodalom

[1] Bruce Schneier,
“Secrets & Lies: Digital Security in a Networked World”,
John Wiley & Sons, 2000.

Taszkok ütemezése desktop-griden

FARKAS ZOLTÁN

MTA Számítástechnikai és Automatizálási Kutató Intézet
zfarkas@sztaki.hu

Lektorált

Kulcsszavak: taszk-ütemezés, skálázhatóság, hierarchikus desktop-gridek

A cikk keretein belül egy viszonylag új grid irányzattal, a desktop gridekkel kapcsolatos taszkütemezés kérdéseit mutatjuk be. A hagyományos gridekkel ellentétben desktop-gridek esetén nem egy adott infrastruktúrába küldi a felhasználó a taszkjait, hanem azok egy központi szerverre kerülnek, ahonnan az erőforrást felajánló donorokon futó kliensek letöltik, majd futtatja azokat. Tehát nem egy taszkhoz keresünk erőforrást, hanem a szabad kapacitással rendelkező erőforrás kér futtatandó taszkokat. A cikk során bemutatjuk a desktop-grideket, pár módszert azok skálázhatóságára, valamint bemutatjuk a hierarchikus desktop-gridekkel kapcsolatos ütemezési kérdéseket és lehetséges algoritmusokat.

1. Bevezetés

A hagyományos, szolgáltatásalapú gridek mellett egy másik grid irányzat mutat jelentős fejlődést: a desktop-gridek. A szolgáltatás alapú gridekkel [1] ellentétben a desktop-grid lényege az asztali számítógépek szabad számítási kapacitásának önkéntes felajánlásában és annak kihasználásában rejlik [2]. Vagyis egy desktop-gridbe bárki beléphet. Azonban a belépés szó értelme ebben az esetben más: míg a szolgáltatásalapú gridek esetén a belépő felhasználók használhatják a rendelkezésre álló infrastruktúrát, addig desktop-gridek esetén a felajánlott számítógépek alkotják az infrastruktúrát. A kihasznált szabad számítási kapacitásokért a felhasználók *krediteket* kapnak. A felajánlott számítógépek természetesen bármikor elhagyhatják a rendszert, ebből adódik a grid-jelleg. További különbség, hogy hagyományos gridek esetén a felhasználók tetszőleges alkalmazást futtathatnak a griden, desktop-gridek esetén a futtatható alkalmazások köre korlátozott: általában egy desktop grid egy probléma megoldására specializálódik, így egy alkalmazást futtat sok különböző paraméterrel.

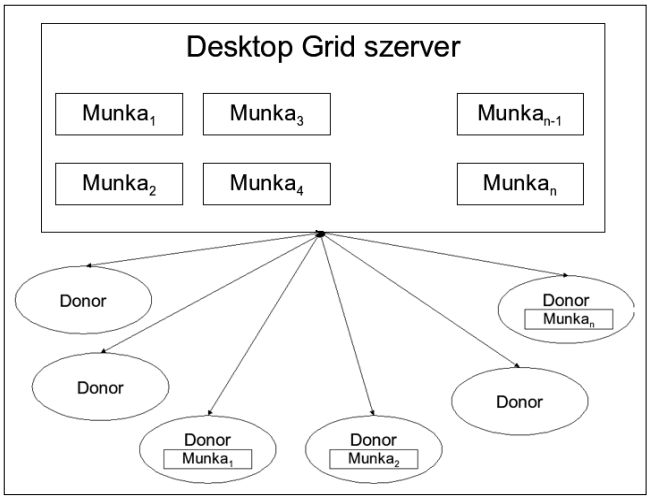
A desktop-gridek ideálisak olyan problémák megoldására, melyeknél egy nagyobb feladatot le tudunk bontani nagyszámú kisebb részfeladatra, ezek eredményéből pedig az eredeti probléma megoldása következtethető (master-worker típusú feladatok). Másik tipikus alkalmazási feladatosztály a parametrikus ellemzések (parameter study alkalmazások), ahol ugyanazt a feladatot kell nagyon sok, akár több tízezer paraméterrel lefuttatni. (Itt jegyezzük meg, hogy az ilyen parametrikus vizsgálatokat szolgáltatói grideken is el lehet végezni [11], vagy például a BME-n kidolgozott Saleve rendszerrel is végrehajthatók [12].

A master-worker típusú feladatok esetén, ha az eredeti probléma megoldása nagyon hoszszú ideig futna egy számítógépen (akár egy klaszteren, akár egy szuperszámítógépen), apróbb feladatokra leosztva viszont a részfeladatok számítási igénye annyira lecsökken, hogy viszonylag rövid idő (néhány óra, esetleg pár nap) alatt feldolgozható egy hagyományos PC-n. Desktop-gridekre példák a következő alkalmazások: közel 250 ország másfél millió számítógépének kapacitását használja a SETI@Home [3], mely a világűrből érkező rádiójelek feldolgozását végzi. Kisebb (bár korántsem elhanyagolható) volumenű projektek még az Einstein@Home [4] és a Climateprediction.net [5]. Hazánkban a nyilvános SZTAKI Desktop Grid (SZDG) futtat hasonlóan BOINC alapú nyilvános projektet, melynek célja a sokdimenziós bináris számrendszerek megtalálása [13]. Az algoritmust az ELTE Komputeralgebra tanszéke fejlesztette ki és az MTA SZTAKI-val közösen adaptálták az SZDG-re.

Az említett projektek közös vonása, hogy BOINC-ra [6] alapozva építették fel az infrastruktúrát. A BOINC a következő elven működik: egy központi szerveren található a projekthez kapcsolódó honlap, a futtatandó alkalmazás(ok) és az alkalmazás(ok)hoz kapcsolódó munkacsomagok. A munkacsomagok kaphatnak prioritást: nagyobb prioritási szint beállításával jelezheti a desktop-grid adminisztrátora, hogy számára az adott munka kiszámolása fontosabb, mint a többié. A felhasználók egy BOINC kliens telepítésével kapcsolódhatnak a szerverhez (így ajánlhatnak fel egy számítógépet – *donort*), ahonnan a kliens letölti a futtatandó alkalmazást, valamint adott mennyiségű munkacsomagot feldolgozásra. Amint van munkacsomag, a BOINC kliens elindítja az alkalmazást, amely feldolgozza a munkacsomagot. A feldolgozás végeztével a BOINC kliens feltölti a számolás eredményét a központi szerverre.

Az *1. ábra* mutatja egy desktop-grid felépítését.

A cikk további részeiben először röviden bemutatjuk az desktop-grid esetén felmerült taszkütemezéssel kapcsolatos kérdéseket és az azokkal foglalkozó cikkeket. Utána bemutatunk három módszert desktop-gridek számítási kapacitásának egyszerű növelésére, majd ezek közül egyet részletesen körüljárunk. Végül ejtünk pár szót a további lehetséges kutatási irányokról, majd röviden összefoglaljuk a leírtakat.



1. ábra A desktop grid felépítése

2. Ütemezési kérdések

A legfontosabb kérdések: mit ütemezzünk és miért, másként: mi az ütemezés célja? Az egyértelmű cél az, hogy a még fel nem dolgozott munkacsomagokat minél előbb kiszámolják a donor számítógépek. Vagyis a munkacsomagokat szeretnénk kiosztani olyan módon, hogy:

– a nagyobb prioritású munkacsomagok előbb kerüljenek feldolgozásra,
– a donorok lehetőleg minél kevesebbet dolgozzanak feleslegesen,
– a desktop-griden található összes munkacsomagot a lehető legrövidebb időn belül kell feldolgozni.

A bevezetés alapján feltehetjük még a kérdést, milyen ütemezéssel kapcsolatos kérdések merülhetnek fel desktop-gridek esetén? Több vonatkozásban beszélhetünk ütemezésről: egyrészt, kérdés, hogy mennyi munkát igényeljen egy donor (pontosabban a donoron futó BOINC kliens) [7,8].

Amikor egy donor munkát kap, a szerver megjegyzi, hogy kinek osztotta ki a feldolgozandó adatot és kiosztáskor egy határidőt rendel a munkacsomaghoz. A megadott határidőn belül vissza kell érkeznie az eredménynek a szerverre, különben a szerver azt feltételezi, hogy a donor valamiért nem tudta befejezni a feldolgozást és kiosztja más donornak a munkát. Tehát lényeges, hogy a donor csak annyi munkát kérjen, amennyit fel is tud dolgozni határidőre. Másrészt, egy donor több desktop-gridhez is kapcsolódhat, így kérdés, hogy az egyes desktop gridek között milyen arányban ossza meg szabad számítási kapacitását. Ezt az arányt a donort felajánló beállíthatja, vagyis ha úgy érzi, hogy két (vagy több) projekt közül számára az egyik valamiért különösen fontos, a szabad számítási kapacitás nagyobb részét ajánlhatja fel számára. Így a határidő betartását szorgalmazó ütemezés tovább bonyolódik: figyelembe kell venni az egyes projektek súlyát is.

Harmadrészt, felmerülhet a kérdés, hogy a szerver végez-e valamilyen ütemezést, azaz mely munkacsomagokat küldi ki először feldolgozásra? A jelenlegi BOINC implementáció elsősorban a munkacsomagok prioritása szerint csökkenő, másodsorban a munkacsomag létrehozási ideje szerint növekvő sorrend alapján osztja ki a munkacsomagokat feldolgozásra.

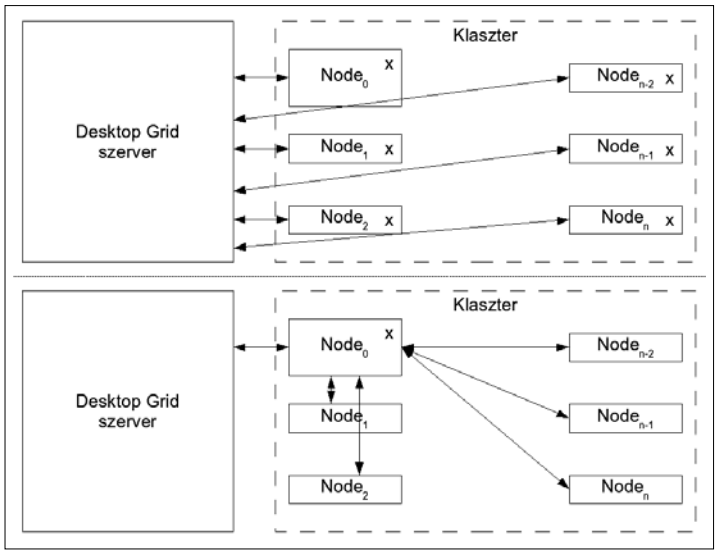
3. Desktop-gridek skálázhatósága

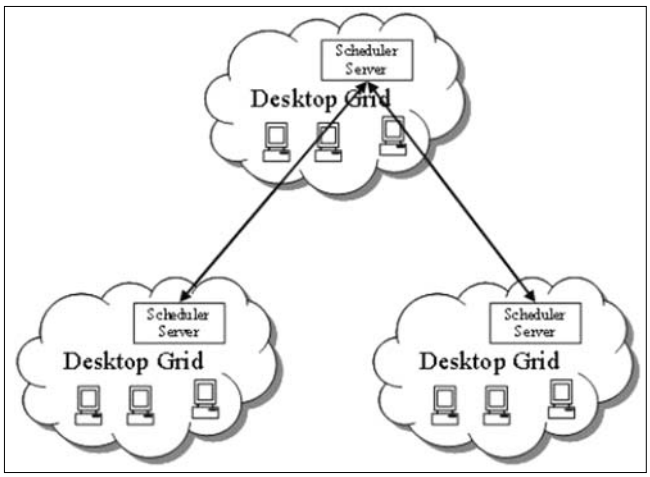
Felmerül a kérdés, meddig növelhető egy desktop-grid teljesítménye sima számítógépekkel, illetve mekkora teher egy-egy új számítógép felajánlása? Mint a bevezetőben láttuk, egy számítógép bekötése pár egyszerű lépést igényel csupán, de számítógépek százainak (pl. klaszter) bekötése már időigényes és monoton munka. A feladat egyszerűbbé tétele érdekében az MTA SZTAKI kifejlesztett egy speciális BOINC klienst, az úgynevezett klaszter klienst [9], mely hatalmas méretű klaszter felajánlása esetén is csupán egyetlen kliens telepítését igényli. A telepítés után a klaszterkliens feladata, hogy a klaszter számítógépeire szétossza a kapott feladatokat. A BOINC szerver szempontjából a klaszer egy többprocesszoros számítógépként látszik és ennek megfelelően is kér munkát a szervertől. A *2. ábrán* ‚X' jelöli a szükséges felajánlásokat klasztergépek egyenkénti, illetve klaszterklienssel végzett felajánlása esetén.

További bővítési lehetőség desktop-gridek számítási kapacitásának növelésére az, ha a desktop-grideket hierarchiába, fastruktúrába szervezzük. A struktúrában alsó szinten levő desktop-grid munkát igényelhet a felette levő desktop-gridtől. Így a hierarchia alkalmazásával lehetőség nyílik arra, hogy egy desktop-grid teljesítményét egy másik desktop-grid teljesítményével növeljük. A hierarchikus modell egy implementációját elkészítette az MTA SZTAKI [10].

A *3. ábra* példát mutat egy hierarchikus desktop-grid rendszerre, ahol a felsőbb szintű desktop-gridtől (például egyetem egy karának desktop-gridjétől) munkát kérnek az alsóbb szintű desktop-gridek (például az egyetemi kar tanszékeinek desktop-gridjei).

2. ábra Lehetséges klaszter-felajánlások





3. ábra Hierarchikus desktop-grid rendszer

Továbbgondolva a hierarchikus desktopgrid-modellt, a fastruktúra mellett lehetőség van egyenrangú desktop-gridek kialakítására, ahol az egyenrangú desktop-gridek tetszés szerint adhatnak át egymás között munkát. Ekkor a fenti ábrán látható alsó szintű desktop-gridek például munkát cserélhetnek egymás közt.

4. Skálázható desktop gridek ütemezési kérdései

Az előző részben három lehetséges módot mutattunk desktop gridek skálázására: klaszterek illesztése, hierarchia kialakítása, illetve egyenrangú desktop gridek összekapcsolása.

Klaszter illesztés esetén a klaszter kliens csupán anynyiban módosul az eredeti klienshez képest, hogy többprocesszoros számítógépként reprezentálja a klasztert. Ebből a szempontból klaszterek esetén a BOINC kliens ütemezési algoritmusa megfelelő, hiszen az fel van készítve több processzoros donorok kezelésére.

Az egyenrangú desktop gridek témaköre egyelőre koncepcionálisan létezik, így ezen rendszeren belüli taszkok ütemezésével a cikk keretein belül nem foglalkozunk.

Ebben a részben a hierarchiába szervezett desktop gridekkel kapcsolatos ütemezések kérdéseit tárgyaljuk részletesebben. Először bemutatjuk a hierarchikus kialakításból származó ütemezési problémákat, majd bemutatjuk azokat az eseményeket, amelyek egy hierarchikus rendszer állapotát (ez által az ütemezési algoritmusok működését) befolyásolják, végül bemutatunk pár ütemezési algoritmust, melyek hierarchikus desktop gridek esetén alkalmazhatóak.

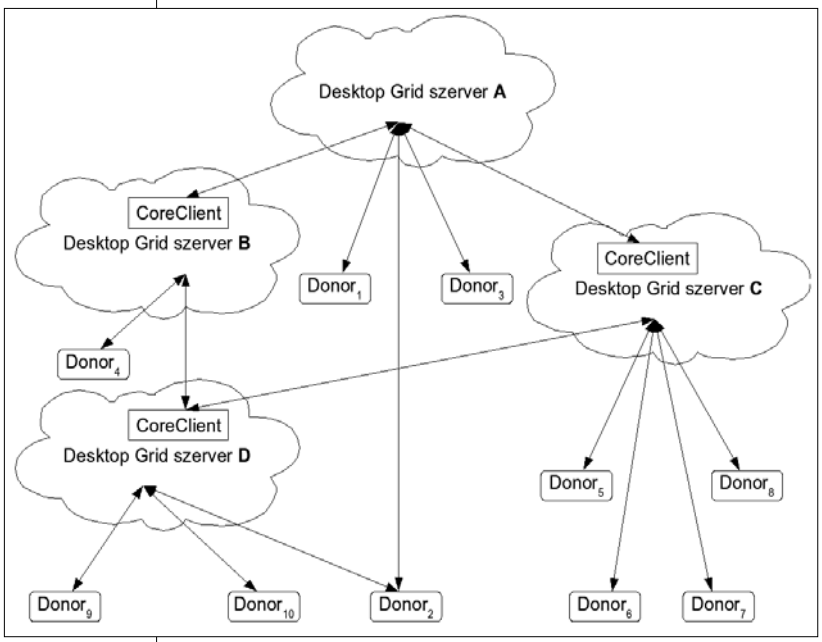
4.1. Hierarchikus desktop gridek ütemezési problémái

Hierarchikus desktop gridek esetén jelentkező ütemezési problémákat legegyszerűbben a *4. ábra* segítségével lehet bemutatni. Az ábra alapján a következő problémákat azonosíthatjuk: túl sok/kevés munka kérése felsőbb szintről, határidő túllépése és felsőbb szintekről származó munkacsomagok közötti különbségtétel.

Az első probléma akkor jelentkezik, amikor egy alsóbb szintű desktop grid által kért munka mennyisége nem tükrözi a desktop grid teljesítményét. Példaként a lenti ábra jelöléseit használva, ha a ‚B' desktop grid kis teljesítményű, de sok munkát kér az ‚A' desktop gridtől, akkor ezzel munkát vonhat el a nagyobb teljesítményű ‚C' desktop gridtől. Ennek ellenkező esete, amikor a sok donorral rendelkező desktop grid kevés munkát kér és a donorok nagy része „malmozik".

A második probléma akkor jelentkezhet, amikor egy desktop grid túlvállalja magát (mert túl sok munkát kér), és a letöltött munkacsomagok határideje lejár a felsőbb szintű desktop griden. Ezt az alsóbb szintű desktop grid egészen addig nem veszi észre, míg meg nem próbálja visszatölteni az eredményt felsőbb szintre. Vagyis, túl sok munka kérése esetén az alsóbb szint donorjai feleslegesen dolgozhatnak.

Hierarchikus desktop gridek esetén kérdés, hogy adjunk-e prioritást a felsőbb szintről érkező csomagoknak, illetve ha egy desktop grid több felső szinttől is kér munkát, melyiket részesítse előnyben, mely csomagjait dolgozza fel előbb? A felsőbb szintű desktop gridtől származó munkacsomagok prioritása könnyen biztosítható, ha az onnan származó munkacsomagok nagyobb prioritással kerülnek be az alsó szintű desktop grid szerverre, mint bármelyik, nem felső szintről származó munkacsomag prioritása.



4. ábra ösztetett hierarchikus rendszer

Hierarchikus rendszerben egy desktop grid több felsőbb szinthez is csatlakozhat. Ebben az esetben alkalmazható a BOINC kliens módszere: az alsóbb szintű desktop grid adminisztrátora megadhatja, hogy a munkacsomagok hány százaléka érkezzen az egyes felsőbb szintű desktop gridektől.

4.2. Hierarchikus desktop grideket befolyásoló események

Számos olyan esemény létezik, amely közvetlenül módosítja egy hierarchikus rendszer állapotát, például: donorok be- és kilépése, új munkacsomag megjelenése, munkacsomag átadása, munkacsomag feldolgozása, desktop grid be- és kilépése.

• Új donor megjelenése egy desktop grid teljesítményét növeli. A donor azonnal kapcsolódik a kérdéses szerverhez és onnan munkát kér, melyen elkezd dolgozni.

• Donor kilépése csökkenti a desktop grid teljesítményét. Sajnos ebben az esetben a desktop grid szerver nem értesül azonnal a kilépés tényéről, vagyis ha a donor kért korábban munkacsomagokat, azokat addig nem osztja ki a szerver újabb donoroknak, amíg azok határideje le nem jár.

• Új munkacsomag megjelenése többféleképpen hathat: ha a munkacsomag prioritása magas, akkor lehetőleg minél előbb meg kell kapnia egy donornak feldolgozásra. Ha nincs prioritása, akkor bekerül a várakozási sor végére.

• Munkacsomag átadás akkor következik be, amikor egy alsóbb szintű desktop grid felsőbb szintről kér munkát. Ekkor felsőbb szinten az átadott munkacsomagokhoz generálódik egy határidő, amin belül eredménynek kell érkeznie, különben feleslegesen dolgoztak az alsóbb szintű desktop grid donorjai.

• Desktop grid belépése többféle módon történhet: ha alsó szinten lép be egy desktop grid, akkor teljesítménynövelő szerepet tölt be. Felsőbb szintre történő belépéskor az alá bekapcsolt desktop grideknek plusz munkát jelent a tőle származó munkacsomagok feldolgozása, vagyis az ő szempontjukból saját csomag-feldolgozási teljesítményük csökken. Köztes szintre is beléphet az új desktop grid, ekkor szerepe kettős: fogad is és továbbít is munkacsomagokat.

• Desktop grid kilépése több dolgot eredményezhet. Ha felső szintű desktop grid lép ki, akkor munka tűnik el. Ekkor, ha alsóbb szint kapott munkát, feleslegesen számolja azt ki. Alsóbb szintű desktop grid kilépése azt eredményezi, hogy a felsőbb szintről kiosztott munkát addig nem kapja meg másik donor (vagy desktop grid), amíg határideje le nem jár.

4.3. Hierarchikus desktop gridek ütemezési algoritmusai

Ebben a részben röviden bemutatunk néhány lehetséges ütemezési algoritmust hierarchikus desktop gridek esetére, majd elemezzük, hogyan reagálnak a korábban bemutatott főbb állapotmódosító eseményekre.

A bemutatásra kerülő ütemezési algoritmusok közös tulajdonsága, hogy „lokális” algoritmusok, vagyis nincs rálátásuk a teljes hierarchikus rendszerre, hatókörük az egyes desktop gridekre korlátozódik, vagyis csak a hozzájuk kapcsolódó desktop gridről rendelkeznek információval, a felsőbb szintű desktop gridek állapotáról információjuk nincs.

4.3.1. Alapütemezés

Ez az „ütemezési” algoritmus a SZTAKI által létrehozott hierarchikus modell implementáció alapértelmezett algoritmusa. Lényege a következő: az alsó szintű desktop grid fix ‚n’ processzorral rendelkező donorként mutatja magát a felsőbb szintek felé, azaz a felsőbb szintekről származó, feldolgozás alatt levő munkacsomagok száma maximum ‚n’ lehet.

Az ütemezés több korábban említett problémát nem old meg: mivel előre kötött a kért munkák száma, ezért az algoritmus nem követi, ha a dekstop grid teljesítménye növekszik, vagy csökken. Így elképzelhető olyan extrém eset (például üres alsó szintű desktop grid esetén), amikor a frissen kapcsolódó donorok nem kapnak munkát. Ellenkező esetben (donorok kilépésekor), amikor a desktop grid teljesítménye csökken, túl sok munkát kér, így a felsőbb szintről származó munkák határideje rendre lejár: a donorok feleslegesen dolgoznak.

Amennyiben a desktop grid adminisztrátora követi a változásokat, módosíthatja a kért munka mennyiségét, de ez külső, emberi beavatkozást igényel.

4.3.2. Donorfüggő ütemezés

Ez az ütemezési algoritmus annyiban próbálja javítani az alap ütemezést, hogy a kért munkacsomagok száma követi az alsó szintű desktop gridhez kapcsolódó donorok számát. Vagyis, új donorok belépése vagy donorok kilépése esetén ez jó megoldás, az algoritmus alkalmazkodik a változáshoz. Donor kilépése esetén fontos, hogy a kilépő donort a felhasználója törölje a rendszerből, különben az algoritmus feltételezi, hogy a donor még mindig dolgozik az alsó szintű desktop grid számára. Vagyis donor kilépés esetén az algoritmus (hasonlóan az alap ütemezéshez) emberi beavatkozást igényel a helyes működéshez.

4.3.3. Aktívdonor ütemezése

A donorfüggő ütemezési algoritmus lehetséges kiegészítése egy olyan szűrés, amely csak az aktív donorok számára kér munkát, azaz az olyan donorok kiszűrése, melyek hosszabb idő óta nem jelentettek le kész munkát. Két lehetőség kínálkozik a passzív donorok szűrésére:

– a desktop grid adminisztrátor adott időközönként törli azokat a donorokat az adatbázisból, melyek az utolsó ellenőrzés óta nem töltöttek fel eredményeket, vagy
– az ütemezési algoritmus valósítja meg a donorok szűrését.

Az emberi beavatkozást elkerülendő célszerű az utóbbi megoldást választani. Kérdés, mikor kell az algoritmusnak egy donort paszszívnak tekintenie? A passzivitás meghatározása a következők alapján történik: az algoritmus passzívnak tekinti azokat a donorokat,

– amelyekhez nem tartozik munkacsomag, így kiszűrhetőek azok a donorok, akik nem kértek újabb munkát, továbbá
– azokat a donorokat, amelyekhez ugyan tartozik munkacsomag, de a munkacsomag feldolgozási határideje lejárt.

A fenti két feltétel vizsgálatával biztosan csak annyi donor számára fog munkát kérni az algoritmus, ahány a fentiek szerint aktív.

Az aktívdonor ütemezés tehát javítja a donorfüggő ütemezést: emberi beavatkozás nélkül képes kiszűrni a passzív donorokat, és az aktív donorok számától függő mennyiségű munkát kér.

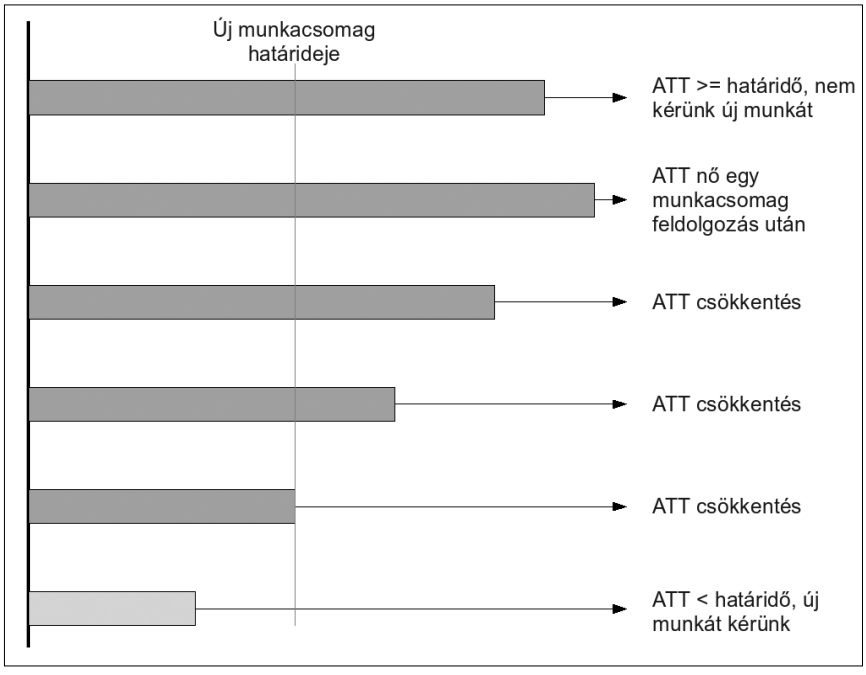
4.3.4. Timeout ütemezés

Az eddig említett alap, donorfüggő és aktívdonor ütemezési algoritmusok mindegyike a desktop grid állapotától, pontosabban a donorok számától függő mennyiségű munkát kért felsőbb szintről (az alap ütemezés bizonyos értelemben kakukktojás, hiszen fix számú munkacsomagot kér). Az említett algoritmusok hiányossága, hogy nem veszik figyelembe a felsőbb szintről kapott munkacsomagok határidejét, vagyis hiába kér annyi munkát, amennyi donor dolgozik, ha a donorok nem tudják teljesíteni a határidőket (esetleg azért, mert a donorok nagyon lassú számítógépek).

A probléma kiküszöbölése érdekében a Timeout algoritmus nyilvántartja a munkacsomagok átlagos megfordulási idejét (*Average Turnaround Time*, ATT): a munkacsomag megfordulási idők (*Turnaround Time*, TT – a munkacsomag felsőbb szintről való letöltése és a hozzá tartozó eredmény visszatöltése közötti idő) átlagát. A timeout ütemezés lényege, hogy az algoritmus folyamatosan frissíti az ATT értékét, azaz statisztikát készít a desktop grid teljesítményéből. Munkacsomag letöltésekor ellenőrzi a kapott munkacsomag határidejét. Amennyiben a kapott határidő kisebb, mint az aktuális ATT, eldobja a munkacsomagot és nem kér több munkát. Így azonban holtpontra juthat az algoritmus: ha nem kap újabb munkát, nincs ami az ATT-t csökkentse. Az ilyen helyzetek elkerülésére az algoritmus adott időközönként csökkenti az ATT értékét. Majd ha az ATT a legutolsó letöltött és eldobott munkacsomag határideje alá csökkent, ismét próbálkozik munka letöltésével.

Az *5. ábra* a timeout algoritmus működését mutatja.

5. ábra ATT csökkentése



4.3.5. Donortimeout algoritmus

A timeout algoritmus problémásan működhet abban az esetben, ha jelentősen eltérő teljesítményű donorok tartoznak a rendszerhez. Tegyük fel, hogy két donor dolgozik az alsóbb szintű desktop grid számára és felső szintről ugyanolyan számítási igényű és határidejű munkacsomagokat kap a desktop grid. Az egyik donor 10 perc alatt végez a munkával, a másik 50 perc alatt, és a munkacsomagok határideje 40 perc. Egyértelmű, hogy a lassú donor sosem fog érdemben végezni egy munkacsomaggal sem, az ATT kezdetben mégis 30 perc, vagyis feleslegesen kér az algoritmus munkát a lassú donor számára is.

A probléma kiküszöbölésére a donortimeout algoritmus donoronként tartja nyilván az ATT értékeket, ezáltal elkerülhető a fentebb említett gond: az első donor ATT értéke 10 perc, számára mindig kér újabb munkát az algoritmus, a lassú donor ATT-je viszont kezdettől fogva 50 perc, vagyis csak azon ritka alkalmakkor fog számára munkát kérni az algoritmus, ha a hozzá tartozó ATT értékét 30 perc alá csökkentette.

4.3.6. Várakozási sor

Az eddig bemutatott algoritmusok figyelték a donorok számát, esetleg figyelembe vették a munkacsomagok átalagos feldolgozási idejét, a kapott munkák határidejének viszonyát. Viszont nem foglalkoznak a desktop grid állapotával, azon belül a helyi, még feldolgozásra váró munkákkal. Abban az esetben, ha a felsőbb szintről kapott munka nem élvez prioritást a helyiekkel szemben, feldolgozásuk csak a helyi munka után történik meg.

A várakozási sor működése során figyelembe veszi az aktív donorok halmazát, a kapott munka prioritását, a kapott munka határidejét, valamint a feldolgozásra váró munkák halmazát. Új munka letöltésekor prioritás szerint sorba rendezi a létező munkacsomagokat és az aktív donorok ismeretében becslést ad az egyes munkák elvégzésének idejére (például pesszimista becslés esetén feltételezi, hogy mindig a leglassabb donorok kapják a munkát). Amennyiben a kapott munka nem dolgozható fel határidőn belül, elveti azt és adott ideig nem is kér munkát felsőbb szintről.

A várakozási sor algoritmus valójában több ütemezést is magába foglalhat: a létező munkák feldolgozási idejének becslési módszerétől függően más és más ütemezési algoritmust kapunk.

5. Összefoglalás

A cikk keretein belül a desktop gridekben található munkák ütemezésének kérdéseit és azok lehetséges megoldásait jártuk körbe. Először bemutattuk a desktop grid fogalmát és megemlítettünk pár olyan eredményt, mely

a munkák donor oldali ütemezésével foglalkozik, vagyis azzal a kérdéssel, hogyan végezhető el adott mennyiségű munka határidőn belül, a szabad számítási kapacitás kihasználásával.

Következő lépésként több módszert is ismertettünk a desktop gridek skálázására: klaszter bevonását, hierarchikus desktop grid építést és az egyenrangú desktop gridek felépítését. A három módszer közül a hierarchikus desktop gridekkel foglalkoztunk részletesebben: megvizsgáltuk, milyen problémákra kell odafigyelni az ütemezés során, bemutattuk mik befolyásolják az ütemezési algoritmusokat, végül példákat mutattunk több lehetséges ütemezési algoritmusra, melyek több szemszögből becsülik meg a letöltendő munka mennyiségét.

További munkák között szerepel az algoritmusok teljesítményének szimuláción keresztüli vizsgálata, valamint az egyenrangú desktop gridekkel kapcsolatos ütemezés körüljárása.

A fenti algoritmusokat az „Új generációs grid technológiák kifejlesztése és meteorológiai alkalmazása a környezetvédelemben és az épületenergetikában” című Jedlik Ányos projekt [14] keretében létrehozandó hierarchikus SZTAKI Desktop Grid rendszerekben fogjuk alkalmazni. A projekten belüli főbb alkalmazási területek: a klíma-modellezés és az épületek hűtéstechnikájának vezérlése. További fontos alkalmazása lesz a kidolgozandó ütemezésnek a CancerGrid EU FP6-os projektben, ahol rák elleni orvosságok fejlesztési idejének csökkentése a cél [15].

A hierarchikus SZTAKI desktop gridek megnyitják az utat a desktop gridek széleskörű és skálázható alkalmazásának irányába a kutatóhelyek és a vállalatok számára egyaránt.

Köszönetnyilvánítás

Jelen cikkben bemutatott munka az NKFP2-00007/2005 projekt keretein belül készült, amit a Nemzeti Kutatási és Technológiai Hivatal tett lehetővé a Jedlik Ányos program keretein belül az új generációs grid technológiák kifejlesztésére és meteorológiai alkalmazására a környezetvédelemben és az épületenergetikában.

Irodalom

[1] I. Foster, C. Kesselman, eds.,
The Grid: Blueprint for a New Computing Infrastructure.
Morgan Kaufmann, San Francisco, 1999.

[2] David P. Anderson:
Public Computing: Reconnecting People to Science,
Conference on Shared Knowledge and the Web.
Residencia de Estudiantes, Madrid, November 2003.

[3] D. P. Anderson, J. Cobb, E. Korpela,
M. Lebofsky, D. Werthimer:
SETI@home: An experiment in public-resource computing. Communications of the ACM,
November 2002, Vol. 45, No.11., pp.56–61.
http://setiathome.berkeley.edu/

[4] http://einstein.phys.uwm.edu/

[5] http://climateprediction.net

[6] David P. Anderson:
BOINC: A System for Public-Resource Computing and Storage.
5th IEEE/ACM Int. Workshop on Grid Computing,
8 November 2004, Pittsburgh, USA.

[7] Derrick Kondo, David P. Anderson, John McLeod VII:
Performance Evaluation of Scheduling Policies for Volunteer Computing.
3rd IEEE Int. Conf. on e-Science and Grid Computing,
Banagalore, India, 10-13 December 2007.

[8] David P. Anderson, John McLeod VII:
Local Scheduling for Volunteer Computing.
Workshop on Large-Scale, Volatile Desktop Grids (PCGrid 2007) held in conjunction with
the IEEE Int. Parallel & Distributed Processing Symposium (IPDPS), 30 March 2007, Long Beach.

[9] P. Kacsuk, N. Podhorszki, T. Kiss:
„Scalable Desktop Grid System”, High performance computing for computational science VECPAR’06,
Rio de Janeiro, Brazil, 10-13 July 2006, pp.1–13.

[10] P. Kacsuk, A. Marosi, J. Kovacs, Z. Balaton,
G. Gombas, G. Vida, A. Kornafeld:
SZTAKI Desktop Grid –
a Hierarchical Desktop Grid System,
Cracow Grid Workshop, Krakow, 2006.

[11] P. Kacsuk, G. Sipos, A. Tóth, Z. Farkas,
G. Kecskeméti, G. Hermann:
Defining and running Parametric Study Applications by the P-GRADE Portal.
Cracow Grid Workshop, Krakow, 2006.

[12] Zs. Molnár, I. Szeberényi:
Saleve: Simple Web-Services Based Environment for Parameter Study Applications,
6th IEEE/ACM Int. Workshop on Grid Computing,
Seattle, 2005.

[13] Burcsi Péter, Gombás Gábor, Kornafeld Ádám,
Dr. Kovács Attila, Kovács József, Marosi Attila Csaba,
Dr. Podhorszki Norbert, Vida Gábor:
„Szuperszámítógépes teljesítmény szuperszámítógép nélkül – A Binsys Projekt”,
Networkshop 2006.

[14] http://www.nkth.gov.hu/main.php?folderID=922

[15] http://www.cancergrid.eu/

Grid Brókerek evolúciója: egységben az erő

KERTÉSZ ATTILA

SZTE Informatikai Tanszékcsoport, Szoftverfejlesztés Tanszék
MTA SZTAKI Párhuzamos és Elosztott Rendszerek Kutatólaboratórium
keratt@inf.u-szeged.hu

Lektorált

Kulcsszavak: Grid Brókerek, erőforrás-választás, Meta-Brókerek, együttműködés

A közel egy évtizede beharangozott, és azóta folyamatosan bővülő és egyre széleskörűbben használt grid rendszerek mára számos európai és világméretű projektek kutatási célpontjaivá váltak. Ezen heterogén, erőteljesen dinamikus rendszerek erőforrásainak eléréséhez és hatékony használatához nélkülözhetetlen, úgynevezett Grid Brókerek, erőforrás-kezelő rendszerek alkalmazása. Az új trendek és fejlődési irányok a brókerek fejlesztésére is kihatottak: kezdetben elegendő volt egy erőforrást társítani egy tetszőleges feladathoz, mára egyre fontosabbá válik a felhasználók igényeinek kiszolgálása. Az üzletiesedő grideken minőségi szolgáltatásokat kell nyújtani és kulcsfontosságú az együttműködés megteremtése az eltérő megvalósítású, de hasonló elven működő grid rendszerek között.

1. Bevezetés

A 90-es években kezdett kibontakozni egy új kutatási irány az elosztott számítások területén, melyet grides számításoknak (Grid Computing) neveztek el. A gridrendszerek lényege a világ különböző tájain lévő számítási rendszerek összekapcsolása, nagyobb számítási kapacitás elérése érdekében. Az érdeklődés egyre nagyobb ezen szakterület iránt; ezt bizonyítja a számos világméretű gridkutatással foglalkozó projekt (CoreGRID [1], LA Grid [2], Globus [3]). Ekkor még a nagy számítási igényű feladatokkal rendelkező kutatók kétkedve néztek a grideket hirdető, népszerűsítő fejlesztőkre, akik rövidebb futtatási időt és kényelmes kezelő-felületet ígértek. Időközben a grid rendszereken belül különféle kihívások megoldása és komponensek fejlesztése végett eltérő kutatási irányok körvonalazódtak ki, melyek önálló kutatási területté emelték a grides számításokat. A gridrendszerek fejlődése során számos kutatási területről (biológia, kémia, fizika) érkeztek felhasználók, akik a kezdeti nehézségek ellenére beléptek a gridet alkalmazók körébe. Ma már a friss statisztikák és kutatási eredmények is azt mutatják, hogy helyesen cselekedtek.

A napjainkra elegendően stabil és megbízható gridek kutatása a felhasználói igényekre összpontosít, hiszen ezen követelmények elengedhetetlenek a majdan üzleti célokat szolgáló gridek számára. A gridrendszereken belül az erőforrás-kezelő komponensek fejlesztésével foglalkozó kutatási területet érinti a leginkább a felhasználói igények felerősödése. Ezek alapján a kutatás két fő igényre összpontosít: a szolgáltatás szintű szerződések lehetővé tételére (Service Level Agreements, WS-Agreements [4]) és az eltérő megvalósítású szolgáltatói gridek együttműködésének elősegítésére. E cikk az utóbbi cél elérésére tett kísérleteket és kutatási irányokat mutatja be a grides erőforrás-kezelés témakörében. Bár napjainkra számos jól megtervezett, széles körben használt grides erőforrás-kezelő rendszer (Resource Management System), grid-bróker [5] elérhető a felhasználói közösség számára, ezek az eszközök a gridet megvalósító, úgynevezett köztes réteg (grid middleware) komponenseire, szolgáltatásaira épülnek, melyek kevéssé adnak lehetőséget az újonnan felmerült igények kielégítésére. A jelenlegi megvalósítások nagy része nem képes átlépni a köztes réteg alkalmazói korlátait, ezáltal a teljes grid rendszer fejlesztésével azonos mértékben fejlődhetnek, mely igen lassú előrelépést és az új igények tekintetében radikális változtatásokat jelent. A több évtizede kidolgozott feladat-ütemezési stratégiák ritkán használhatók grides környezetben, aminek a gyakran változó terheltség és elérhetőség, valamint a grid middleware korlátjai jelentenek akadályt. Mindezek ellenére hazánkban is folynak kutatások újabb, a grides környezethez alkalmazkodó ütemezési stratégiák kidolgozására [14]. Emellett napjaink szolgáltatói gridjei viszonylag elkülönített felhasználói közösséggel és fejlesztői csoporttal rendelkeznek, mely szintén az együttműködés elősegítésének útjában áll.

Az *1. ábrán* látható napjaink grides alkalmazása: a grides erőforrások elérése általában grid portálokon keresztül történik, de lehetőség van közvetlenül az erőforrás-brókerek meghívására is.

Az együttműködő gridek problémájával nagytekintélyű szakértői csoportok is foglalkoznak. Az egyik ilyen, Európában irányadó grides szakértői csoport a Next Generation Grids Expert Group, mely az Európai Bizottság égisze alatt működik. Legújabb közleményükben [6] az európai gridek jövőjéről, a 2010-ig megvalósítandó és azon túlmutató célokat, kutatási irányokat jelölték ki. Ebben a webes és grides technológiák konvergenciáját állapították meg és egyben kijelölték az utat a szolgáltatás-orientált tudás-alapú komponensek, SOKU-k (Service Oriented Knowledge Utility) fejlesztése felé, melyeknek együttműködő, megbízható és hibatűrő működést megvalósító megfelelő tudás-bázissal rendelkező szolgáltatásoknak kell lenniük.

Mindezen felhasználói igényeket és szakértői útmutatásokat figyelembe véve ez a cikk egy olyan magas szintű erőforrás-kezelő szolgáltatást javasol az együttműködési probléma megoldására, mely az előírásoknak megfelelő tulajdonságokkal rendelkezik és nem igényli a köztes réteg komponenseinek újratervezését.

2. Erőforrás-kezelés és választás – a kezdetektől napjainkig

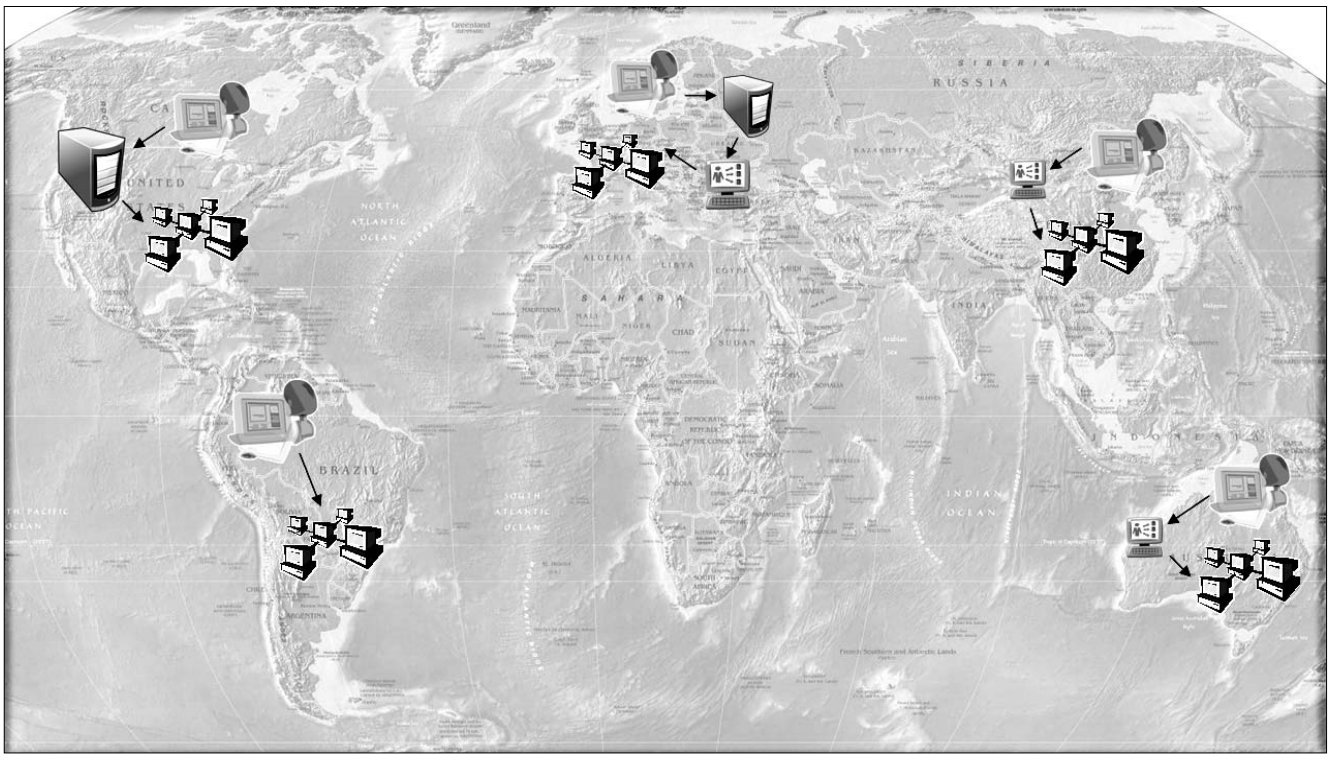
Kezdetben a grides erőforrás-kezelő komponensek csak a rendszerbe bekapcsolt, számításokat végző számítógépek elérését tették lehetővé egy feladatot beküldő, állapotot ellenőrző és eredményt letöltő interfészen keresztül. Az egyre nagyobb méretű és egyre több feladatot kiszolgáló gridekben viszont felmerült az igény olyan brókerek kifejlesztésére, melyek a gridek információs rendszerével kommunikálva a feladat végrehajtásának legmegfelelőbb (általában lehető leggyorsabb futási időt biztosító) erőforrásra ütemezze a felhasználó feladatát (job-ját). Az erőforrás-információk begyűjtésén és az ütemezésen túl a Grid Bróker feladata az erőforrással történő kapcsolatfelvétel, a feladat beküldése, állapotának ellenőrzése és jelentése a felhasználónak, végül az előállított eredmény visszajuttatása. A felhasználói feladatok leírása feladat-leíró nyelven (job description language) történik, ezt kell átadni a brókernek a futtatható állománnyal együtt.

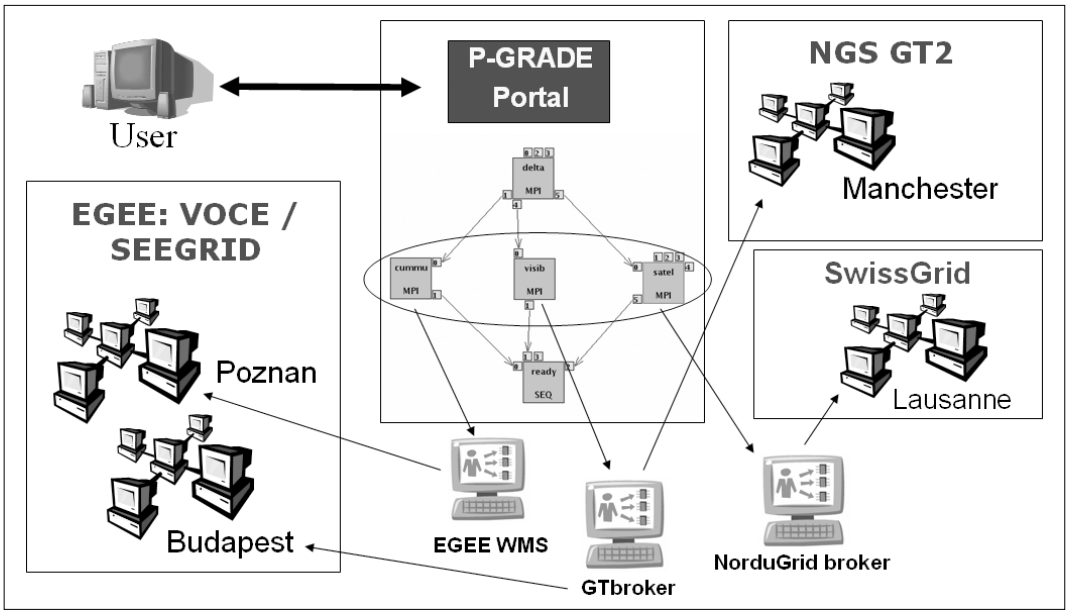
Itt szembesülünk az első problémával: a különböző megvalósítású gridrendszerek általában eltérő formátumú leíró nyelveket használnak. Ezen túlmenően, bár a köztes réteg komponensei és szolgáltatásai azonos működési elvvel rendelkeznek, szintén eltérő protokollt használnak a fájlok továbbítására, az információs rendszer adatainak tárolására és elérésére, valamint az erőforrások kezelésére. Ezen eltérések láttán nem meglepő, hogy mind a felhasználók (a leíró nyelvek formátuma miatt), mind a fejlesztők (az eltérő protokollok és interfészek miatt) csoportokba tömörültek és az általuk választott rendszer használatát, illetve fejlesztését választották. Ugyanezen érvek miatt az egyes Grid Brókerek is bizonyos köztes réteghez, gridrendszerhez kötöttek.

Napjainkra a kutatók és bróker-fejlesztők többsége felismerte ezt a problémát, és megtették az első lépést az együttműködés elősegítése érdekében: elkezdték újratervezni, kibővíteni a brókerek bizonyos komponenseit lehetővé téve újabb nyelvek megértését és eltérő protokollok használatát. Ennek köszönhetően egyes brókerek már képesek többféle griddel együttműködni, ezáltal különböző felhasználói csoportokat kiszolgálni. Ezek a bővítések napjainkban is folynak, viszont érthető módon hosszabb időt és több munkát igényelnek, nem is beszélve arról, hogy ezáltal a brókerek egyre összetettebbé és megbízhatatlanabbá válnak, hiszen nő a hibalehetőségek száma.

A másik viszonylag könnyen megvalósítható megoldás a gridportálok kibővítése. Ezek az eszközök kényelmes, egyszerűen használható felhasználói felületet biztosítanak az egyes gridkomponensek használatára, a felhasználói feladatok végrehajtására. Léteznek korlátozott szolgáltatásokkal rendelkező vagy bizonyos feladatok támogatására specializált portálok is. Erre egy példa a hazánkban kifejlesztett Conflet keretrendszer (CONFigurable portLET) [13], melynek segítségével egyedi feladatokhoz készíthetünk portált, megkönnyítve ez-

1. ábra Szolgáltatói gridek felhasználása napjainkban





2. ábra
Multi-grid elérés és brókerezés a P-GRADE Portálban

zel a grides alkalmazást, futtatást. Bár a legtöbb grid portál egy adott gridrendszerhez kötött, lehetséges a brókerek kibővítéséhez hasonlóan újabb nyelvek, protokollok és interfészek támogatásának beépítése, ezáltal a portál képes lesz több szolgáltatói gridet használni, eltérő felhasználói közösségeket kiszolgálni. További előny a nagyobb számítási kapacitás biztosítása, hiszen több grid több erőforrást jelent.

A *2. ábra* mindkét megoldásra bemutat egy példát a P-GRADE Portál [7] használatán keresztül. Ez a gridportál egy általános workflow-fejlesztő és -futtató környezetet nyújt a felhasználók számára. Az ábrán a Portál dobozban a nagyobb téglalapokkal jelölt elemek a végrehajtandó felhasználói programok (delta, cummu stb.), ezek összekapcsolásával áll elő a grides alkalmazás, más néven workflow. Az alkalmazás megszerkesztése során a felhasználó nyilakkal kötheti össze a futtatható programok kimeneti és bemeneti fájljait (a kisebb, számozott téglalapok), ezek a nyilak az egyes programok közötti függőségeket jelölik. A szerkesztés utolsó fázisában erőforrásokat vagy brókereket rendelhetünk a programokhoz (más néven feladatokhoz, jobokhoz), melyek az alkalmazás indítása után elvégzik a feladatok futtatását a hozzájuk tartozó gridekben. A megoldás hátránya hasonló, mint az előző esetben: eltérő gridek támogatása a rendszer módosítását, újratervezését igényli.

3. Evolúciós lépés: Grid Brókerek egyesítése

Az előzőekben bemutatott nehézségek láttán több kutatói csoport is új utat keresett az együttműködési probléma megoldására. Nyilvánvalóvá vált, hogy a jelenlegi architektúra megtartásával a közeljövőben nem, csak a gridek köztes rétegének hosszabb időt igénylő fejlődése után valósítható meg az együttműködő gridek világa. A megoldás kulcsa a brókerek közötti kommunikáció megvalósításában rejlik.

Az egyik legnagyobb világméretű grid kutatói szervezet az OGF (Open Grid Forum) a grid rendszerek minden területén szabványosításra és új megoldások kidolgozására törekszik. Számos kutatási csoportja közül az egyik, az OGF-GSA-RG (Grid Scheduling Architecture Research Group [8]) egy minden bróker által elfogadott és megvalósított interfész kidolgozását tűzte ki célul. Egy ilyen interfész lehetővé tenné, hogy kommunikáljanak egymással a brókerek és különféle felhasználói feladatokat osszanak meg egymás között, közösen válasszanak erőforrást a feladatoknak, mindezáltal a lehető legnagyobb számítási kapacitást érhetnék el a felhasználók. Az első nagy nehézséget jelentő probléma ebben a megközelítésben a közös interfész kidolgozása. Ha ez a közeljövőben meg is születne, az egyes brókerek interfészhez igazítása, újratervezése, majdan a közös ütemezési művelet megtervezése és megvalósítása biztosan hosszú időt fog igénybe venni.

A másik megközelítés az elkülönített rendszerekben működő hasonló megvalósítású brókerpéldányok kommunikációját célozza meg. Mivel ezen brókerpéldányok megegyeznek és ugyanazon kutatók fejlesztik, egyszerűbbé válik közös interfész megalkotása és használata. (Megjegyzendő, hogy ez esetben az eltérő kutatói csoportok vélhetően más-más interfészen dolgoznak, ennél fogva a teljes grid rendszer ismételten szeparált marad.) Ezt az irányvonalat követik a következő projektek: Koala [9], LA Grid [2] és Gridway [10].

Mindegyik megoldás egy-egy saját brókerpéldányt működtet elkülönített gridekben vagy eltérő virtuális szervezetekben, domain-ekben. A brókerpéldányok képesek kommunikálni egymással és ha a saját hatáskörükben lévő erőforrások túlterheltek vagy nem képesek végrehajtani a felhasználói feladatot, továbbítják azt egy másik domain-t kiszolgáló példányhoz. A szerzők legutóbbi publikációikban már bemutatták, hogy járható az általuk kijelölt irány, ugyanakkor az eltérő rendszerek közötti együttműködés még ebben az esetben sem megoldott.

A végső megoldást az úgynevezett meta-brokering, a létező brókereket együttesen használó, magas szintű, metaadatokat felhasználó brókerezés jelenti. Ez a megközelítés egy újabb szintet hoz létre a gridbrókerek fölé és azokat együttesen használja a felhasználók kiszolgálására. Az OGF korábban kidolgozott egy szabványt a feladatleíró nyelvek egységesítésére, ez lett a JSDL (Job Submission Description Language [11]).

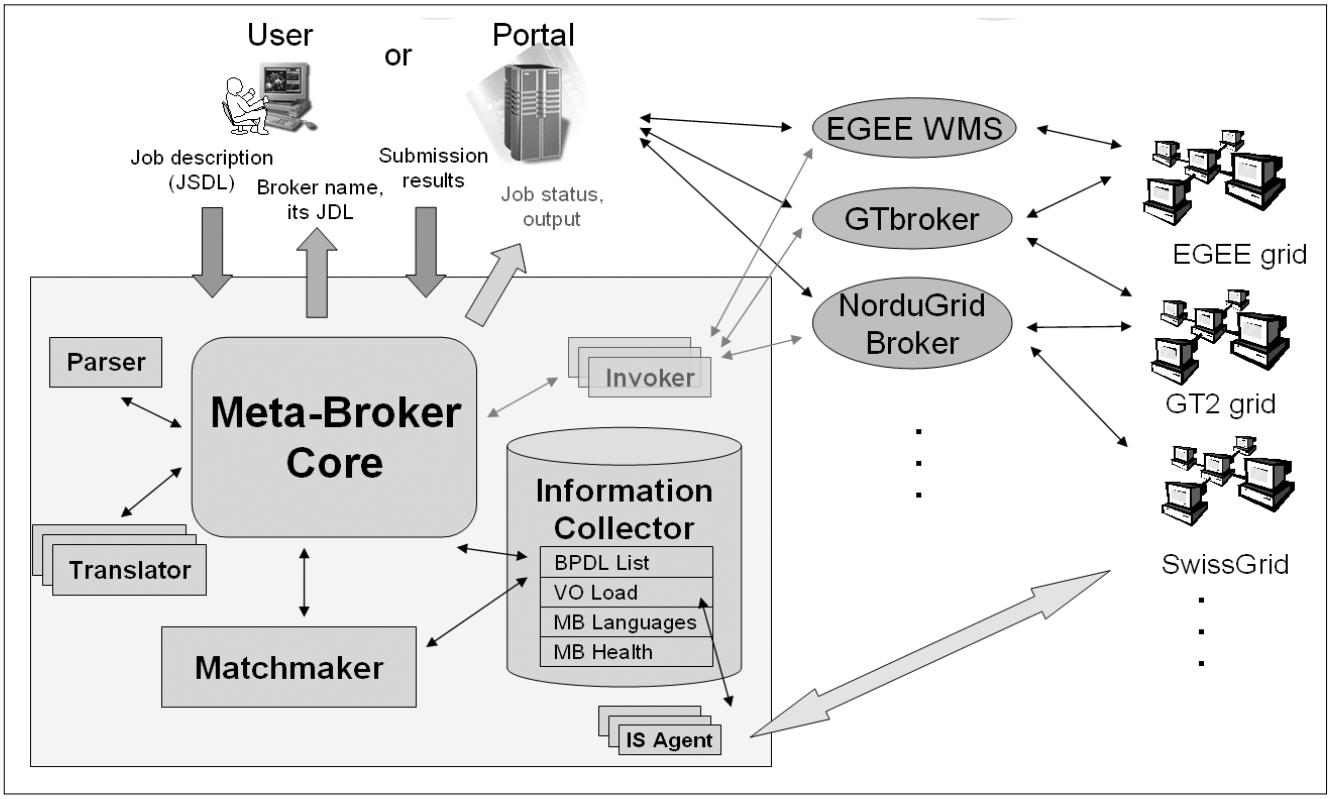
A Grid Meta-Bróker tervezése során definiáltunk egy új nyelvet a brókerek tulajdonságainak egységes leírására, azaz a brókerekkel kapcsolatos metaadatok tárolására – ezt neveztük el BPDL-nek (Broker Property Description Language [12]). A meta-brókeres szinten történő ütemezéshez szükség van még egyéb ütemezési felhasználói igények leírására és minőségi szolgáltatást biztosító szintén ütemezéssel kapcsolatos brókertulajdonságok definiálására.

Ezeket az attribútumokat a BPDL első verziója tartalmazta, a legújabb, folyamatban lévő fejlesztésünkben viszont külön választottuk ezeket az adatokat és mind a felhasználói feladatok, mind a brókerek leírására használhatjuk azokat a JSDL és a BPDL kiegészítéseként. Ez az újabb nyelv az MBSDL (Meta-Broker Scheduling Description Language) nevet kapta, mellyel hangsúlyozzuk, hogy a nyelv attribútumai ütemezési tulajdonságokat és igényeket írnak le. A szétválasztásra azért volt szükség, mert mind a BPDL mind a Meta-Bróker JSDL kiegészítő nyelve tartalmazta ezen attribútumok jelentős részét – így fölöslegesen, duplán tároltunk bizonyos adatokat.

A másik indok a szabványosítás megkönnyítésében rejlik: az OGF-GSA-RG [8] már korábban elkezdett egy ütemezéssel kapcsolatos leíró nyelvet kidolgozni (SDL, Scheduling Description Language). Mivel az SDL követelményrendszerét lefedték a BPDL 1.0 és a JSDL kiterjesztés bizonyos részei, kigyűjtöttük ezeket az attribútumokat az MBSDL nyelvbe és felvettük a kapcsolatot a kutatócsoporttal a kompatibilitás megtartása és a további munka elősegítése céljából. Az ütemezési algoritmus ezeket a metaadatokat használja fel arra, hogy válasszon az adott feladathoz egy brókert, ezáltal egy gridet, futtatási környezetet. A kiválasztás után a feladatot eljuttatja a brókernek, amely végrehajtja az általa megszokott módon. A Meta-Bróker feladata az általános feladatleírás lefordítása a választott bróker nyelvére.

A Grid Meta-Bróker architektúrát *(3. ábra)* egy önálló webszolgáltatásként valósítottuk meg. Ezáltal független a grides köztes rétegektől és szabványos interfészeken keresztül kommunikál a felhasználók és a gridek felé. A brókerválasztást IS Agent-ek, grid-információs rendszerekkel kapcsolatban álló ügynökök segítik. Az általuk gyűjtött információkkal kiszűrhetőek a túlterhelt vagy hibásan működő erőforrásokat használó brókerek. Az Invoker nevű komponensek a Meta-Bróker által használt brókerek meghívását, a feladatok továbbítását és az eredmények begyűjtését végzik. Egy másik lehetséges megvalósításban az Invoker-ek használata helyett a felhasználóra vagy a Meta-Brókert használó portálra bízhatjuk a tényleges feladatbeküldést. Ez esetben a Meta-Bróker a kiválasztott bróker nevét és a lefordított leírást adja vissza a felhasználónak és a feladat lefutása után értesítést vár a lefutás sikerességéről. Ezt az információt az adott brókert leíró teljesítményadatokban frissíti.

3. ábra Grid Meta-Bróker architektúra



4. Összefoglalás és jövőbeli tervek

Láthattuk, hogy a bevezetésben említett új kihívásoknak nem képesek megfelelni a korábban megalkotott grides erőforrás-kezelő rendszerek. A legújabb felhasználói igényeket, nagyobb számítási kapacitást és üzleti követelményeket kiszolgáló szolgáltatás megvalósításához egy új szemléletre, magasabb szintű megközelítésre van szükség: egyesíteni kell az elkülönült szolgáltatói grideket menedzselő brókereket. Ezen brókerek további rendszerekhez történő adaptálása vagy portálokba integrálása jó kezdeti megoldásnak bizonyul, de hosszú távon kezelhetetlenné és sérülékennyé válnának.

Egy másik ígéretes megközelítés az azonos megvalósítású bróker példányok egymás közötti kommunikációjának lehetővé tétele, mely az azonos brókerek által működtetett gridek együttműködését megoldja, viszont ezek az egyesített gridek ugyancsak elkülönülnek egymástól, hiszen az eltérő fejlesztésű brókerek nem képesek kommunikálni egymással.

A végső megoldást a Grid Meta-Bróker alkalmazása jelenti, mely a legújabb elvárásoknak megfelelő felépítéssel rendelkezik és szabványos nyelveket és interfészeket használ az együttműködő gridek megvalósításához. A meta-brókeres működés lényege, hogy a már jól bevált, széles felhasználói körrel rendelkező brókereket együttesen alkalmazva egy közös hozzáférési szolgáltatást nyújtsunk az összes felhasználó számára az összes elérhető gridhez. A bemutatott Meta-Bróker megvalósítása folyamatban van, a következő lépés a rendszer tesztelése, mellyel bizonyítjuk a hatékonyabb, kényelmesebb és egyszerűbb grid alkalmazást.

Egy világméretű egyesített grid rendszerben (WWG, World Wide Grid [15]) a Meta-Brókerek játszhatják majd az összekötő szerepet, ami szükségessé teszi, hogy az egyes példányok megosszák a feladatokat és kommunikáljanak egymással. A Meta-Bróker architektúra megfelel az NGG jelentésben [6] foglalt követelményeknek, és együttesen alkalmazza a legújabb webes és grides technológiákat.

Köszönetnyilvánítás

A cikkben ismertetett munka a CoreGRID NoE IST-2002-004265 projekt támogatásával jött létre.

Irodalom

[1] http://www.coregrid.net
[2] http://www.latinamericangrid.org/
[3] http://www.globus.org/
[4] http://www.ogf.org/documents/GFD.107.pdf
[5] A. Kertész, P. Kacsuk: A Taxonomy of Grid Resource Brokers, 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS 2006) in conjunction with the Austrian Grid Symposium 2006, Innsbruck, 21-23 September 2006, pp.201–210.
[6] Next Generation Grids Report: Future for European Grids: GRIDs and Service Oriented Knowledge Utilities – Vision and Research Directions 2010 and Beyond, (NGG3), December 2006.
[7] A. Kertész, G. Sipos, P. Kacsuk: Multi-Grid Brokering with the P-GRADE Portal, In post-proceedings of the Austrian Grid Symposium (AGS'06), OCG Verlag, Austria, 2007.
[8] https://forge.gridforum.org/sf/projects/gsa-rg
[9] A. Iosup, D. H.J. Epema, T. Tannenbaum, M. Farrellee, M. Livny: Inter-Operating Grids through Delegated MatchMaking, In proceedings of the Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC07), Reno, Nevada, November 2007.
[10] T. Vazquez, E. Huedo, R. S. Montero, I. M. Llorente: Evaluation of a Utility Computing Model Based on the Federation of Grid Infrastructures, (Euro-Par 2007), 28 August 2007. pp.372–381.
[11] http://www.ogf.org/documents/GFD.56.pdf
[12] A. Kertész, I. Rodero, F. Guim: Data Model for Describing Grid Resource Broker Capabilities, CoreGRID Workshop on Grid Middleware in conjunction with ISC'07 Conference, Dresden, Germany, 25-26 June 2007.
[13] D. Pasztuhov, I. Szeberényi: A Conflet rendszer új architektúrája, Networkshop 2007. https://nws.niif.hu/ncd2007/docs/ehu/036.pdf
[14] L. Cs. Lőrincz, A. Ulbert, Z. Horváth, T. Kozsik: Towards an Agent Integrated Speculative Scheduling Service, 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS 2006), Innsbruck, 21-23 September 2006., pp.211–222.
[15] P. Kacsuk, A. Kertesz, T. Kiss: Can We Connect Existing Production Grids into a World Wide Grid?, Submitted to 8th International Meeting High Performance Computing for Computational Science (VECPAR'08), Toulouse, France, 24-27 June 2008.

Grid rendszerek használata vasbeton hídgerendák tervezésében

PASZTUHOV DÁNIEL, SZEBERÉNYI IMRE
BME Irányítástechnika és Informatika Tanszék
{dani, szebi}@iit.bme.hu
SIPOS ANDRÁS ÁRPÁD
BME Szilárdságtani és Tartószerkezeti Tanszék
siposa@silver.szt.bme.hu

Lektorált

Kulcsszavak: grid, portál, ipari alkalmazás, vasbeton gerendák, párhuzamos számítás

Cikkünkben bemutatunk egy valós, mérnöki feladatot és annak megoldását egy hatékony párhuzamos algoritmussal, valamint egy – felhasználói felületek előállítását hatékonyabbá tevő – webes eszközt is, melynek segítségével kényelmes, webes felületen indíthatók párhuzamos és grid-feladatok, valamint lehetőség van parancssoros programok indítására is.

1. Bevezetés

Bár a grid nagy számítási kapacitása miatt az ipari alkalmazások széles körében alkalmazható lenne, a használat nehézkessége eddig meggátolta az ilyen jellegű felhasználás elterjedését. Cikkünkben bemutatunk egy hatékony párhuzamos algoritmust, amellyel egy mérnöki probléma megoldása számítható. Az algoritmus alkalmas arra, hogy a számításokat grid-környezetben hajtsuk végre. A hozzá tartozó felhasználói felület lehetővé teszi az ipari felhasználók számára az eljárás egyszerű használatát. Segítségével lehetőség van a szükséges adatok bevitelére, azok fájlba történő elmentésére, korábbi adatfájlok beolvasására és természetesen a számítás elindítására, valamint az eredmények letöltésére is. A felületet létrehozó konfigurálható portlet egy újonnan kifejlesztett nyelvet használ a felület működésének leírására. A keretrendszer alkalmas arra, hogy a parancssoros adatbeviteltől a grafikai megoldásokig az alkalmazások széles körét biztosítsa a felhasználói felület számára. Megközelítésünk konfigurálhatóság szempontjából eltér a Grid környezetekhez fejlesztett portálrendszertől (Genius [16,17], P-GRADE [18], BIRN [19], LEAD [20]).

A párhuzamos algoritmus előfeszített vasbetongerendák, elsősorban hídgerendák térbeli alakváltozásainak számítására szolgál. A probléma erősen nemlineáris jellege – a geometriai és anyagi nemlinearitás egyszerre befolyásolja az egyensúlyi alakot – ellenére az eljárás megbízható, nem kell tartani hamis megoldásoktól vagy divergens viselkedéstől. A szakirodalomban [5-7] eddig javasolt eljárások bizonyítottan vezethetnek hamis megoldásra, vagy produkálhatnak kaotikus viselkedést. Módszerünk megbízhatóságának ára a nagy számításigény, hiszen pontos eredményhez mintegy 1 millió rúdalakot kell kiszámítani.

Cikkünk második fejezetében röviden ismertetjük az algoritmust, kiemelve a vasbeton rudak deformációinak számításában rejlő nehézségeket, az eljárás részletes ismertetése több publikációban is megtalálható [1-3]. A harmadik fejezet a felhasználói felülettel szemben támasztott követelményeket tartalmazza, a negyedik fejezetben pedig az új Conflet keretrendszert mutatjuk be, mely alkalmas más eljárások hasonló igényeinek gyors és hatékony kielégítésére.

2. Vasbeton hídgerendák térbeli deformációjának számítása

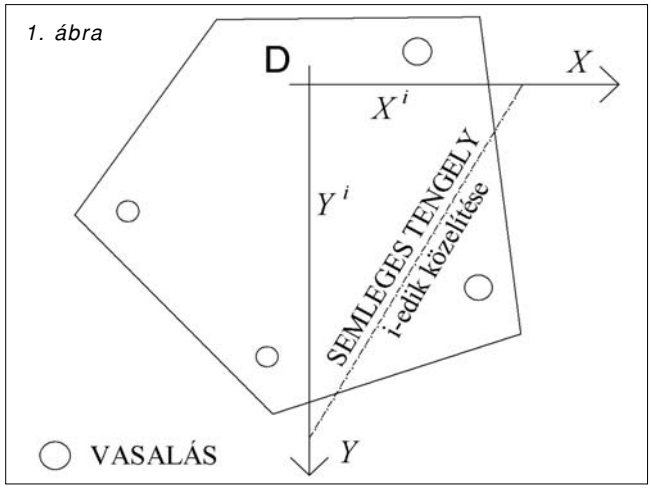
A bevezetőben említett algoritmus vasbetongerendák és -oszlopok térbeli deformációit határozza meg az anyag és a geometria nemlineáris viselkedésének figyelembe vételével. A számítás során a nyíró- és a normálerőkből származó alakváltozásokat elhanyagoljuk, továbbá feltesszük, hogy a sík keresztmetszetek a deformáció után is síkok maradnak.

A rúd azon tartományát, ahol csak nyomófeszültségek ébredhetnek, *betonnak* nevezzük, azon tartományokat pedig, ahol húzófeszültségek is ébredhetnek, *acélnak* hívjuk. A keresztmetszet alakja és a vasak helyzete tetszőleges *(1. ábra)*.

2.1. A keresztmetszet görbületének számítása

A sík keresztmetszetek törvénye miatt a hajlítás hatására a keresztmetszet egy egyenes, a semleges tengely körül fordul el. Célunk a semleges tengely, a gör-

1. ábra



bület és az elcsavarodás számítása. A rúd egy keresztmetszetének igénybevétele tipikusan külpontos nyomás és egyidejű csavarás. A semleges tengely meghatározása egy nem-lineáris egyenletrendszer megoldását követeli meg, hiszen a húzott oldalon a beton bereped, vagyis a dolgozó keresztmetszetet az ismeretlen semleges tengely határolja. Egy, az egyensúlyi egyenletekből származtatott kétdimenziós leképzéssel a semleges tengely helye kevés, 10-15 iterációs lépésen belül egyértelműen meghatározható. A lineáris anyagtörvényhez tartozó leképzés származtatásáról, illetve a konvergenciájának bizonyításáról [1,3,4] számol be részletesen. A beton valóságos viselkedését sokkal pontosabban modellező, fellágyuló anyagtörvényhez is definiálható egy konvergens iteráció, mely egyértelműen meghatározza a semleges tengely helyét és a görbületet.

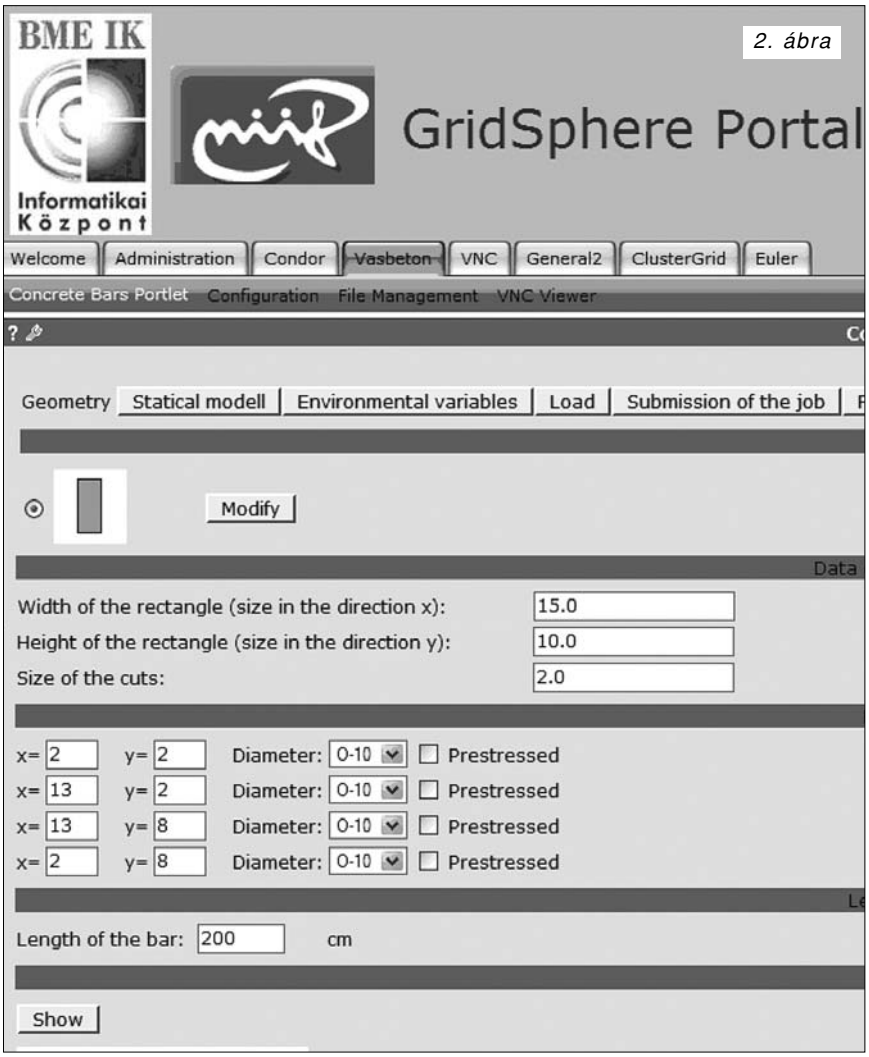
Vasbeton rudak számításánál nem csupán a beton anyagtörvényének nemlinearitását kell figyelembe venni, hanem a beton zsugorodását és kúszását is. A zsugorodási többletgörbületet az EUROCODE szabvány (EC2) [15] alapján közvetlenül lehet számítani. A kúszást a kúszási tényező segítségével számított effektív rugalmassági modulussal vesszük figyelembe. Pontosabb számításokhoz lehetőség van az úgynevezett Trost-féle eljárás használatára. A kúszás és a zsugorodás számítása az algoritmus megbízhatóságát nem befolyásolja. Az előfeszítés kapcsán a megengedhető feszítőerőt, illetve a feszültség-veszteségeket szintén az EC2 szabvány szerint számítjuk.

2.2. A rúd alakjának számítása

A rúd deformációit a görbület és az elcsavarodás hosszmenti integrálásával kaphatjuk meg, amennyiben a rúd egyik végének térbeli helyzetét és az ott működő erőket és nyomatékokat ismerjük. Ebben az esetben *kezdetiérték-feladatról* beszélünk. A mérnöki gyakorlatban tipikusan a rúd mindkét végén vannak ismeretlen geometriai vagy statikai mennyiségek. Ekkor egy *peremérték-feladatot* kell megoldanunk, mely jóval nagyobb számítási kapacitást igényel, mint egy kezdeti-érték feladat kiszámítása.

A peremérték-feladat megoldására kézenfekvő numerikus eljárás a szimplex-módszer [4,8,9], amely nagyszámú kezdetiérték-feladat megoldására vezeti vissza a problémát. Egyetlen rúdból álló szerkezetek esetén változók a rúd egyik végén a peremfeltételek által nem rögzített alakváltozási jellemzőket *változóknak* nevezzük. A rúd másik végén a peremfeltételek által előírt mennyiségeket *függvényeknek* nevezzük. A megoldásokat a változók és a teherparaméter d=n+1 dimenziós terében keressük, amelyet a továbbiakban a probléma *Globális Reprezentációs Terének* (GRS) nevezünk. A GRS egy-egy pontja tehát egy-egy kezdetiérték-feladatnak felel meg és ezek között keressük a vizsgált mechanikai probléma peremérték-feladatának megoldásait. A szimplex módszernél első lépésben a GRS-t diszkretizáljuk egy szimplex-hálóval. (A fenti példa 2D-s GRS-ében ez háromszögekre való felosztást jelent.) Ezután kiintegráljuk a kezdetiérték-feladatot a szimplexek összes csúcspontjában, meghatározva a függvények e pontbeli értékét. A szimplexek belsejében lineáris interpolációt alkalmazva keressük a feladat megoldásait, azon pontokat, ahol minden függvény értéke zérus. Itt tehát egy egyszerű lineáris egyenletrendszert oldunk meg.

A példa teljes megoldásához a GRS összes szimplexében el kell végezni a számítást, ami rendkívül munkaigényes, a számítási igény GRS dimenziójával exponenciálisan nő. Ezért érdemes a módszert nagy teljesítményű, párhuzamos, illetve Grid technológiájú rendszerekre implementálni. A szimplex módszer különösen alkalmas erre, mivel a számításokat szimplexenként külön, egymástól függetlenül lehet végezni, tehát a számítási munka kis, önálló egységekre bontható.



2. ábra

3. A felhasználói felülettel szemben támasztott követelmények

Ahhoz, hogy egy „valós életből” vett gerendát számolni tudjunk, megközelítőleg 200-1000 paraméter megadására van szükség. A paraméterek száma alapvetően a keresztmetszet bonyolultságától függ *(2. ábra)*. A bevitt koordináta-adatoknak megfelelő alakzatot felhasználói ellenőrzés céljából a képernyőn meg kell jeleníteni.

Az anyagi jellemzők megfelelnek az EC2 szabványnak, azaz a felhasználónak választhat a különféle minőségű betonok, acélok és előfeszítő pászmák közül. A felhasználói felület a megfelelő anyagjellemzőket a kiválasztott minőséghez rendeli hozzá. További paraméterek adják meg környezeti jellemzőket (például relatív páratartalom, a szerkezet életkora megterheléskor stb.) Szükséges továbbá a szerkezet terheinek (koncentrált és megosztó terhelést) bevitele.

Ha az összes paramétert megadtuk, a feladat elindítható, vagy a paraméterkészlet fájlba menthető, hogy onnan visszaolvashassuk egy későbbi feladatindítás érdekében. Indítás előtt a Globális Reprezentációs Tér adatai módosíthatóak.

4. A Conflet keretrendszer

A felhasználói felület a Conflet (CONFigurable portLET – konfigurálható portlet) rendszerrel készült. A Conflet rendszer egy egyszerűen használható keretrendszer az alkalmazásfejlesztők számára, hogy segítségével felhasználóbarát felületeket készítsenek grid-környezetben futó feladatok indításához, lecsökkentve ezzel a fejlesztésre fordított időt.

A rendszer arra való, hogy oldalakat, oldalcsoportokat jelenítsen meg és beolvassa róla a feladat különböző paramétereit, elvégezzen egyszerű számításokat (ideértve egy kép generálását), fájlokat hozzon létre és végül grides feladatot (vagy parancssoros távoli programot) indítson. Sok ilyen típusú feladat létezik, így a Conflet fő célkitűzése, hogy minimalizálja a fejlesztő által elvégzett munkát.

A Conflet a nyílt forráskódú, ingyenes GridSphere Portál Keretrendszer [10,11] szolgáltatásaira épít, de olyan rugalmasra lett tervezve, hogy bármely más (nem feltétlenül portál, de mindenképpen Java alapú) keretrendszerrel együtt tudjon működni.

Egy Conflettel létrehozott feladatindító alkalmazás futás közben is megváltoztatható a konfigurációs fájlok (azaz azon fájlok, amik az alkalmazás kinézetét és viselkedését befolyásolják) egy halmazának feltöltésével. A két fő fájltípust *view*-nak és *controller*-nek nevezzük. Egy view határozza meg egy oldal kinézetét, megvalósítását tekintve JSP fájl. A controller határozza meg egy oldal viselkedését, azaz azon akciókat, melyeket a portlet végrehajt egy gomb megnyomására vagy egy link meghívására. Mindkét fájltípusból több példány létezhet és ezen példányok nincsenek összekötve sem: az aktuális view megváltozhat az aktuális controller megváltoztatása nélkül és fordítva is, valamint egy view több controllerhez is tartozhat és egy controller több view-nak is lehet a párja.

4.1. Controller és View

A controllerek megalkotásához létrehoztunk egy XML-alapú vezérlő nyelvet. A controller nyelv string változókat és egyszerű vezérlési szerkezeteket (elágazás, ciklus) használ.

Egy Conflettel előállított felhasználói felületben több view (Java Server Pages (JSP) [14] és GridSphere UI Tab Library) segítségével létrehozott oldal is létezhet. A különféle oldalakat csoportosíthatjuk, úgynevezett fülekbe szervezhetjük. A fülek oldalon belüli pozíciója egy saját jelölővel adható meg a Conflet Tag Library-ből. Egy példa controller és view látható a *3. és 4. ábra*. Az utóbbi a 2. ábra középső részének leírása.

A view és controller fájlok főbb jellemzői a következők:

- Karakterlánc típusú változók definiálhatók és használhatók a view és controller fájlokban. A változók könnyen összefűzhetők egymással vagy egy konstanssal.
- Az űrlapelemek értéke a rendszerben változóként jelennek meg, míg tulajdonságaik controller parancsokkal állíthatók be. Az űrlapelemek értékei fájlba menthetők, és visszatölthetők onnan.

3. ábra

```
<?xml version="1.0"?>
<actions version="1.0.1"
  xmlns:condor="http://n0.iit.bme.hu/gridsphere/condor.xsd"
  xmlns:ssh="http://n0.iit.bme.hu/gridsphere/ssh.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="leiro.xsd">
  <init>
    <for var="i" begin="1" end="${rf_n}">
      <listbox beanId="rf_${i}_dia" file="rfdia.dat"
selectedid="6"/>
      <checkbox beanId="psrf_${i}" selected="false"/>
    </for>
  </init>
  <default>
    <if>
      <or-each var="i" begin="1" end="${rf_n}">
        <string string="${psrf_${i}}"/>
      </or-each>
      <then>
        <enable-tab tabid="prestress"/>
      </then>
      <else>
        <disable-tab tabid="prestress"/>
      </else>
    </if>
  </default>
  <action name="show">
    <generate-postfix var="pngpostfix"/>
    <function
class="hu.bme.iit.gridsphere.vasbeton.geometria.Geometria"
jar="vasbeton.jar" output="cache/geomkep${pngpostfix}.png">
      <input>
        <line>pic</line>
        <line>rectangle</line>
        <line-each var="i" begin="1" end="${paramnum}">
          ${param_${i}}
        </line-each>
      </input>
    </function>
    <image beanId="ready"
src=".${user_name}/cache/geomkep${pngpostfix}.png"/>
  </action>
  <action name="change">
    <next view="geomselect.jsp" ctl="geomselect.xml"/>
  </action>
  <action name="next">
    <next/>
  </action>
</actions>
```

```
<ui:frame>
  <ui:tablerow header="true">
    <ui:tablecell>
      <conflet:message key="DATA_OF_CROSS_SECTION"/>
    </ui:tablecell>
  </ui:tablerow>
</ui:frame>
<ui:frame>
  <ui:tablerow>
    <ui:tablecell width="350">
      <ui:hiddenfield beanId="paramnum" value="3"/>
      <conflet:message key="WIDTH_OF_RECTANGLE"/>:
    </ui:tablecell>
    <ui:tablecell>
      <ui:textfield beanId="param_1" value="15.0"/>
    </ui:tablecell>
  </ui:tablerow>
  <ui:tablerow>
    <ui:tablecell width="350">
      <conflet:message key="HEIGHT_OF_RECTANGLE"/>:
    </ui:tablecell>
    <ui:tablecell>
      <ui:textfield beanId="param_2" value="10.0"/>
    </ui:tablecell>
  </ui:tablerow>
  <ui:tablerow>
    <ui:tablecell width="350">
      <conflet:message key="SIZE_OF_CUTS"/>:
    </ui:tablecell>
    <ui:tablecell>
      <ui:textfield beanId="param_3" value="0.0"/>
    </ui:tablecell>
      </ui:tablerow>
</ui:frame>
```

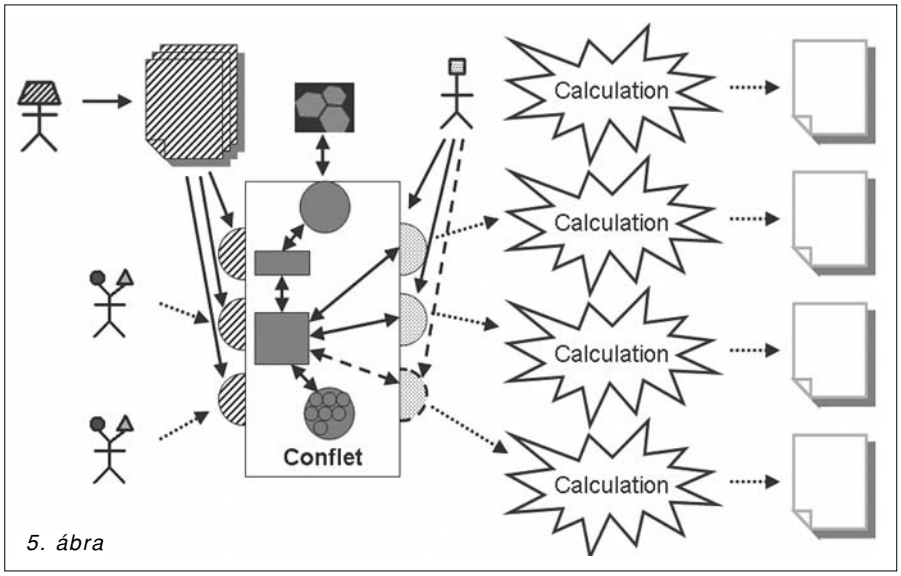
4. ábra

- Fájlok hozhatók létre, tölthetők le és fel a távoli gépre vagy akár az alkalmazásszerver gépére. A fájlok tartalma reguláris kifejezésekkel változókba tehető. A fájl feldolgozásához és írásához ciklusok definiálhatók.
- Névvel ellátott változócsoportok hozhatók létre és tárolhatók XML-fájlokban. A csoportok nevei listadobozba tölthetők, míg a kiválasztott nevű csoportba tartozó változók egy utasítással aktiválhatók.
- A controllerhez Javaban kiterjesztéseket írhatunk, melyek segítségével egyszerű számításokat végezhetünk el, vagy akár képeket hozhatunk létre. A futtatáshoz a Java Security Managert használjuk. A kiterjesztések az archívumnév (JAR), az osztály neve, valamint a ki- és bemenet megadásával hívhatók meg.
- Akciók (parancssorozatok) rendelhetők nemcsak egyes konkrét felhasználói eseményekhez, hanem akár egy oldal első betöltődéséhez (init), vagy egy tetszőleges akció lefutásához (default). Az akciók egymásból is meghívhatók.
- A rendszerhez beépülő modulok (plugin) segítségével új middleware-t, új storage-et és új, távoli helyen parancsot futtatni képes modult (accessor) tudunk adni. A beépülő modulok új parancsokat is tartalmazhatnak.
- A parancsnyelv alapparancsai is könnyedén kiterjeszthetők. Az alapparancsokat tartalmazó modul (Conflet Language Bundle, CLB) könnyedén lecserélhető, tartalma különféle adminisztrációs fájlok módosítása nélkül megváltoztatható. A CLB nem csak új alapparancsokat, hanem új konfigurációs fájltípusokat is tartalmazhat, így a rendszer – bizonyos keretek közt – rugalmasan bővíthető.
- A controllerben lévő hibák a Java stack trace-hez hasonlóan jelennek meg.

4.2. Architektúra

A Conflet többrétegű architektúrát használ, melyet a lenti, *5. ábra* ábrázol.

- A *user interface* modul (a Conflet téglalapján belül a legfelső, szürke kör) köti össze a portál motort (vagy más, megjelenítéshez használt Java környezetet) a rendszer központi moduljával a „Conflet interface” modulon keresztül. Feladata, hogy a fő modulból származó oldalakat megjelenítse, és a felhasználói interakciókat a fő modulhoz visszairányítsa.
- A *Conflet interface* modul (a Conflet téglalapján belül a felső lapos téglalap) definiálja azokat az interfészeket és absztrakt osztályokat, melyek a „user interface” modul és a fő modul közötti interakcióhoz használhatók. (Nem tekintjük külön rétegnek).
- A *fő modul* (a Conflet téglalapján belüli négyzet) keretszolgáltatásokat nyújt a CLB-nek, úgy mint parancs- és konfigurációs fájl-osztályok betöltése, fájlok betöltése, az adatok kezelése, kommunikáció és a CLB elemeinek meghívása.
- A *Conflet Language Bundle* (a Conflet téglalapján belüli kis körökkel teli kör) tulajdonképpen parancsosztályok és konfigurációs fájl-osztályok halmaza kiegészítve néhány segédosztállyal, melyek a CLB koherenciáját hivatottak biztosítani. A controller parancsait a CLB osztályai valósítják

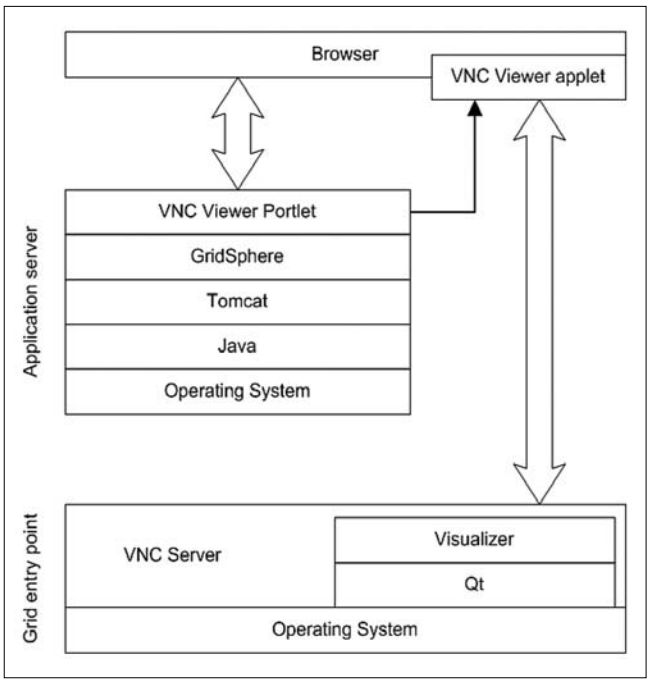


5. ábra

meg, a konfigurációs fájl-osztályok pedig az egyes típusok (például view, controller stb.) működését biztosítják.
- *Plugin modulok* (a Conflet téglalapjának jobb oldalán lévő félkörök) definiálhatók, hogy hozzáférést biztosítsunk a különféle külső szolgáltatásokhoz, mint a grid, egy cluster, távoli storage-ek, vagy program-végrehajtók. A Plugin modulokban további parancsok lehetnek.

A ferdén csíkos részek a konfiguráció menetét jelölik. A trapézfejű pálcikaember (konfigurátor) elkészíti a konfigurációs fájlokat, melyek egy-egy felületként jelennek meg a felhasználó szemszögéből.

A pöttyözött vonal a feladat indításának menetét szemlélteti: a kétfejű felhasználó kapcsolatba lép a Conflet egyik felületével, majd az – a beépülő modulokon keresztül – elindítja a feladatot, ami vizualizálható eredményt ad.



6. ábra

5. Megjelenítés

Az eredmények megjelenítéséhez két feladatot kellett megoldani. Egyrészt egy szoftvereszközt kellett kifejleszteni, amely képes megjeleníteni grafikusan a számítás eredményét, másrészt pedig a kifejlesztett szoftvereszközt integrálni kellett a portál architektúrájába. A vizualizáció architektúrája a *6. ábrán* látható.

5.1. Az eredmények megjelenítéséhez használt alkalmazás

A számolást végző algoritmus eredménye a Globális Reprezentációs Térben a bifurkációs diagram, amit a megjelenítő eszköz két- vagy háromdimenziós formában megjelenít. Ha kiválasztjuk a diagram egy pontját, az alkalmazás kirajzolja a hozzá tartozó rúdalakot, valamint megadja a maximális vízszintes és függőleges lehajlást és elfordulást. Ez azt jelenti, hogy a ponthoz tartozó egyetlen kezdetiérték-feladatot a munkaállomás megoldja, ehhez nem szükséges a párhuzamos környezet.

Az alkalmazást a platformfüggetlen Qt [12] keretrendszer segítségével implementáltuk. A megjelenítő bármely szabványos X-Window vagy Windows környezetben képes futni. Mivel a futtatásra szolgáló távoli gép általában szabványos UNIX környezet, kézenfekvő a programot ott futtatni. Alternatíva, hogy a fájlokat letöltve a felhasználó a saját munkaállomásán jeleníti meg az eredményeket. Előbbi esetben megoldandó probléma volt, hogy miként jelenítsük meg a távoli gép képét a felhasználó számítógépén úgy, hogy lehetőleg ne kelljen semmiféle programot telepítenie. A probléma megoldására fejlesztettük ki a VNC Viewer portletet.

5.2. VNC Viewer portlet

A VNC (Virtual Network Computing) [13,14] lehetővé teszi, hogy a felhasználó grafikus kapcsolatba lépjen bármely géppel az interneten. Egy általános célú grafikus felhasználói felületet biztosít állapotmentes protokoll fölött. A szoftvercsomag két komponensből áll: a szerver a távoli gépen fut, fenntartja a kapcsolatot a kliens és a grafikus környezet között, vagy grafikus szolgáltatásokat nyújt; a kliens (viewer) pedig megjeleníti a munkakörnyezet képét a felhasználó képernyőjén. A kliensnek különféle verziói léteznek, van többek között Java applet verziója is, ami szempontunkból a legfontosabb változat; ezt integráltunk a portálunk környezetébe.

Az elkészült portáloldal a következő szolgáltatásokat nyújtja:
- A távoli gépen VNC munkaasztalok (szerverek) hozhatók létre és törölhetők, valamint csatlakozhatunk hozzájuk.
- Úgynevezett program profilok hozhatók létre, melyek a későbbiekben lehetővé teszik, hogy a távoli munkaasztalunkon mindössze három egérkattintással új programot indíthassunk.

Biztonsági szempontok miatt a VNC jelszó alapú azonosítást használ. Ahhoz, hogy az egyszeres bejelentkezés (single sign-on) követelményeit teljesíteni tudjuk, ezt a tulajdonságot a biztonság szem előtt tartása mellett át kellett alakítanunk, lehetővé kellett tennünk, hogy a felhasználók újabb jelszó beírása nélkül tudjanak VNC szerverükhöz kapcsolódni úgy, hogy azt más az interneten ne tehesse meg.

6. Összefoglalás

Cikkünkben bemutattunk egy globálisan konvergens algoritmust vasbeton hídgerendák térbeli deformációjának számítására. Az algoritmushoz illeszkedő felhasználói felület lehetővé teszi a Grid rendszerek ipari alkalmazását. Bemutattunk továbbá egy, az eredmények megjelenítésére alkalmas eszközt is.

A fejlesztő szemszögéből tekintve a Java/J2EE, XML és GridSphere Portál Keretrendszer technológiákon alapuló alkalmazásfejlesztési rendszer segítségével új fel-

használói felületeket lehet készíteni parancssoros programokhoz a bináris módosítása nélkül. Grid környezetben hasznos eszköz lehet feladatindító portletek létrehozására. Mivel magasszintű parancsokból álló parancsnyelvet használ, a fejlesztés gyorsabbá és hatékonyabbá válik, ráadásul nem kell Java-ban programozni, fordítani és adott esetben szervert újra indítani.

Cikkünkben ismertettük, hogyan definiálhatók felhasználói felületek a kifejlesztett keretrendszer segítségével.

Köszönetnyilvánítás

A cikkben ismertetett munka részben az OTKA TO46646, TS49885 számú, a Nemzeti Kutatás-fejlesztési Iroda NKFP 2/009/04, valamint a Pázmány Péter program RET-06/2005 projektjeinek támogatásával jött létre. A szerzők köszönik az EU INFSO-50883 projekt és a BVM Épelem Kft támogatását.

Irodalom

[1] Sipos, A. A., Domokos G.:
„Asymmetrical, spatial deformations of reinforced concrete columns and prestressed beams”,
fib Symposium „Keep Concrete Attractive”,
Budapest, 2005., Vol. II, pp.693–698.

[2] Sipos, A. A., Domokos G., Gáspár Zs.:
„A 2D Pelikan iteráció konvergencia-tulajdonságai”,
J. of Building Science, 2005, 33 (1-2), pp.205–217.

[3] Sipos, A. A.:
„Calculation of the spatial deformations of rods without tensile strentgh”,
PhD thesis, BME, 2007.

[4] Domokos, G.:
„Global description of elastic bars”,
Zeitschrift für Angew. Math. und Mech., 1994, No.74, T289-T291.

[5] Brondum-Nielsen, T.:
„Stress Analysis of Concrete Sections Under Service Load”,
ACI Journal, Proceedings, 1979, Vol. 76., No.2, pp.195–211.

[6] Brondum-Nielsen, T.:
„Serviceability Limit State Analysis of Cracked, Polygonal Concrete Sections Under Biaxial or Symmetric Bending”,
ACI Journal, Proceedings, 1986, Vol. 83., No.2, pp.209–218.

[7] Cosenza, E., Debenardi, P. G.:
„Calculation of Stresses, Deformations and Deflections of Reinforced and Prestressed Concrete Elements in Service”,
CEB Bulletin 235, 1997, pp.105–142.

[8] Gáspár, Zs., Domokos, G., Szeberényi, I.:
„A parallel algorithm for the global computation of elastic bar structures”,
Comput. Assist. Mech. Eng. Science, 1997, No.4, pp.55–68.

[9] Domokos G., Szeberényi I.:
„A Hybrid Parallel Approach to One-parameter Nonlinear Boundary Value Problems”,
Comput. Assist. Mech. Eng. Science, 2004, No.11, pp.15–34.

[10] M. Russell, J. Novotny, O. Wehrens,
„GridSphere: An Advanced Portal Framework”,
GridSphere Project Website (www.gridsphere.org)

[11] M. Russell, J. Novotny, O. Wehrens,
„GridSphere: A Portal Framework for Building Collaborations”,
GridSphere Project Website (www.gridsphere.org)

[12] Qt 4.0 Whitepaper,
http://www.trolltech.com/pdf/whitepapers/qt40-whitepaper-a4.pdf

[13] RealVNC Home Page,
http://www.realvnc.com

[14] JavaServer Pages 2.0 Specification,
http://jcp.org/aboutJava/communityprocess/final/jsr152

[15] EUROCODE EN 1992-1-1. April 2002, Rev. final draft.

[16] Genius Portal,
https://genius.ct.infn.it/

[17] Genius Portal,
http://egee.cesnet.cz/en/user/genius.html

[18] P. Kacsuk, G. Sipos
„Multi-Grid, Multi-User Workflows in the P-GRADE Portal”.
Journal of Grid Computing, Vol. 3., Issue 3-4, Kluwer Academic Publisher, 2006.
pp. 221–238.

[19] BIRN Portal,
https://portal.nbirn.net/

[20] LEAD Portal,
https://portal.leadproject.org/

Saleve: párhuzamos grid-alkalmazások fejlesztőeszköze

DÓBÉ PÉTER, KÁPOLNAI RICHÁRD, SZEBERÉNYI IMRE

BME Irányítástechnika és Informatika Tanszék
{dobe, kapolnai, szebi}@iit.bme.hu

Lektorált

Kulcsszavak: Saleve, grid alkalmazásfejlesztés, paraméterelemzés, EGEE

A Saleve rendszer egy párhuzamosan futó, úgynevezett paraméterelemző alkalmazások fejlesztését és futtatását segítő eszköz. A Saleve-vel fejlesztett programokat változtatás nélkül integrálhatjuk különböző köztesrétegeket alkalmazó gridekbe, így az alkalmazás fejlesztőjének nem szükséges a köztesréteg technikai részleteit ismernie.

1. Bevezetés

Napjainkban a tudományos számítások futtatására már számos számítógépes erőforrás rendelkezésre áll, melyek használatára azonban még nincs mindenütt megfelelő támogatás. Bár sok fejlesztés célozta meg ennek segítését, egy kutatónak még ma is számos akadállyal kell szembenéznie, ha igénybe kíván venni egy elosztott, párhuzamos számítási rendszert, így például egy gridet.

A fenti nehézségeket kívánjuk áthidalni a Saleve keretrendszerrel, amely elfedi a párhuzamos alkalmazás fejlesztője elől a mögöttes infrastruktúra technikai részleteit. A Saleve egy speciális, könnyen párhuzamosítható feladatcsoport, a paraméterelemző algoritmusok implementálására összpontosít. Segítségével olyan párhuzamos alkalmazások készíthetőek, melyek akár többféle infrastruktúrán is végrehajthatóak változtatás nélkül.

A továbbiakban részletesen ismertetjük a Saleve által támogatott feladatokat és bemutatjuk Európa legkiterjedtebb grides projektjét, az EGEE projektet. Ezt követően felvázoljuk a Saleve fejlesztésének főbb mozgatórugóit és célkitűzéseit, majd a 4. szakaszban bemutatjuk a rendszer működését, végül pedig összefoglaljuk eredményeinket és a továbbfejlesztési lehetőségeket.

2. Grid alkalmazása paraméterelemző feladatokhoz

2.1. Paraméterelemző feladatok

A gyakorlatban sokszor felmerül az igény arra, hogy egy adott algoritmust több száz, vagy akár sok ezer különböző bemeneti paraméterértékkel lefuttassunk: az ilyen feladatok a paraméterelemzések (Parameter Study/ Parameter Scan).

A várt megoldás bizonyos esetekben az egyes paraméterekkel kapott kimenetek összessége, de gyakran egy végső, összesítő lépés révén kapjuk ezekből a végeredményt. Egy kimerítő optimumkeresés során például ez az utolsó lépés egyetlen paraméter kikeresését jelenti. Egy másik egyszerű példa egy nem analitikus függvény numerikus integrálása egy adott tartományon. A tartományt feloszthatjuk egymást nem fedő résztartományokra, ezek lesznek az integrálást elvégző eljárás bemeneti paraméterei, a végső lépés pedig a részintegráloknak az összegzése.

A paraméterelemzés problémaköre felbukkan számos kísérleti tudományágban, különösen fizikai szimulációknál. Ide tartozik továbbá a nagyenergiájú részecskefizika, az asztrofizika, génkutatás, gyógyszerkutatás, földrengés-kutatás. A paraméterelemzéshez hasonlóan részfeladatokra osztható feladat általában minden olyan mérnöki feladat, probléma, amely közönséges differenciálegyenlet-rendszerrel írható le. Ilyenek például a statikai feladatok, mely területen megemlítendő a BME-n végzett kutatás, melynek keretében vasbeton hídgerendák tervezési feladatainál alkalmaznak jól párhuzamosítható, a paraméterelemzéshez részben hasonló algoritmust [1].

A rendkívül nagy számú különböző bemenettel történő futtatásokat időben egymás után végrehajtani azonban igen sok időt venne igénybe. Megállapítható viszont, hogy az egyes lefutások között semmiféle csatolás nincs, így ezek futási sorrendje tetszőleges, ráadásul egymással párhuzamosan is történhet a Single Program, Multiple Data (SPMD) modellnek megfelelően [2]. Több processzor jelenléte tehát jól kihasználható, akár egy multiprocesszoros számítógépről, akár egy sok gépből álló fürtről van szó – sőt, legjobb esetben egy grid infrastruktúra szolgáltatásait is igénybe vehetjük erre a célra.

2.2. A Grid-rendszerek jelene

A rohamosan növekvő számítási és adattárolási igény kielégítése érdekében már több mint egy évtizede felmerült az elképzelés egy földrajzilag elosztott erőforrás-hálózat, grid kiépítésére [3]. Ennek megvalósítására azóta számos kezdeményezés indult a világ különböző területein.

Ezek közül Európában egyik legnagyobb az Enabling Grids for E-sciencE (EGEE) [4]. A svájci CERN kutatóközpontnál levő Large Hadron Collider (LHC) részecskegyorsító érzékelőiből származó adatok feldolgozására kezdték kialakítani a rendszert, mára azonban szá-

mos szerteágazó tudományterületen vannak alkalmazásai: többek között asztrofizikában, bioinformatikában és geofizikában. A projekt 240 intézményt fog össze a világ 45 országából, köztük Magyarországról is. A grid jelenleg körülbelül 41 ezer processzort tartalmaz, 5 petabájt adatot képes tárolni és 100 ezer feladatot dolgoz fel egyidejűleg.

A BME is részt vesz az EGEE-ben, egyrészt szervező tevékenységekkel, másrészt erőforrások rendelkezésre bocsátásával. Az általunk kialakított erőforráskészletben, amely a BMEGrid nevet kapta, jelenleg 8 darab négymagos szerver futtatja a gridbe beküldött programokat. A rendszerhez csatlakoztattunk ezen kívül egy nagy kapacitású és hatékony párhuzamos elérésű adattároló eszközt, a Scalable File Share-t, amely mintegy 3 terabájt adat tárolására képes. Erőforrásainkat többségében a részecskefizikával foglalkozó Atlas kutatócsoport tagjai, valamint biomedikus kutatók használják.

A grid projektek hangsúlyt fektetnek az új alkalmazások fejlesztésének könnyítésére, hogy növeljék a felhasználók táborát. Erre megoldást nyújtanak a portálrendszerek [1,10], helyenként kiegészítve alkalmazásfejlesztési és munkafolyamat-kezelő eszközökkel [11], vagy egyéb, összetett funkciókat is támogató környezetek [12]. A Saleve megközelítése ezektől eltér: segítségével olyan párhuzamos alkalmazások hozhatók létre, amelyek saját magukat képesek a futtató környezethez eljuttatni külön eszköz nélkül és emellett könnyűsúlyúak és egy személyi számítógépen is futtathatóak. A rendszer – mint ahogy az említett környezetek is – független az alkalmazási területtől.

3. A Saleve rendszer áttekintése

3.1 Motiváció

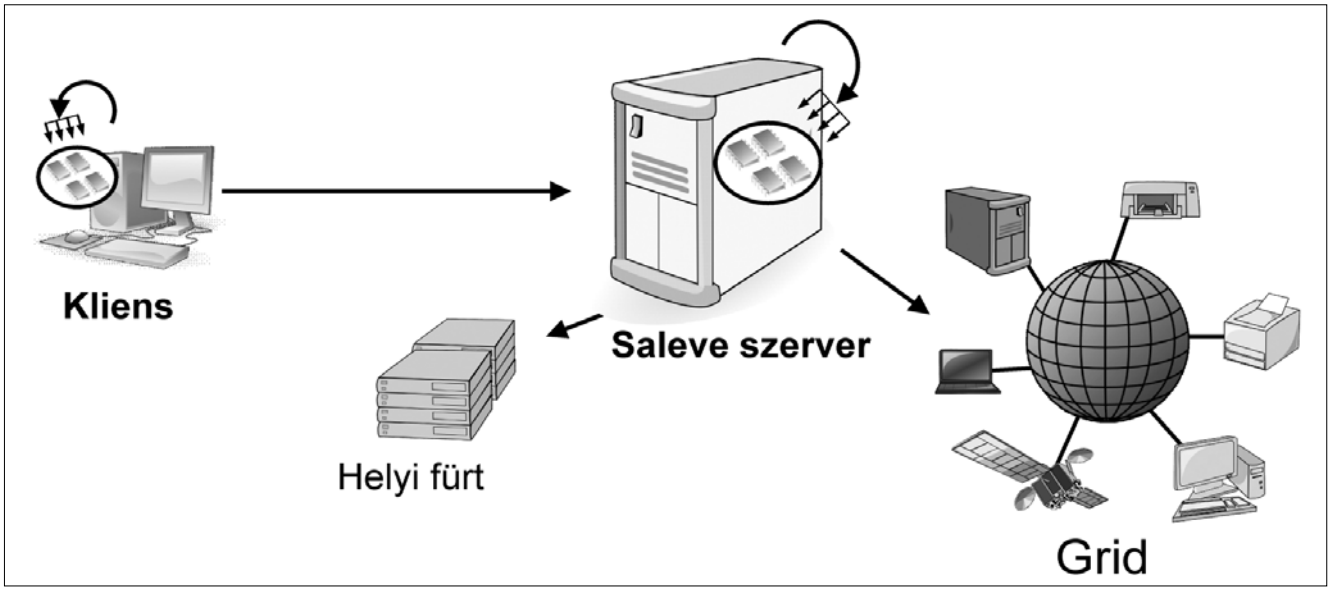
A kutatók, mérnökök nagy része rendelkezik programozási ismeretekkel, különösen a C és Fortran nyelv terén, így a tudományos számításokat végző hagyományos, soros feldolgozású programok elkészítése nem okoz nekik gondot. Párhuzamosan futó, elosztott programok fejlesztése azonban nagyobb programozási tudást és gyakorlatot igényel. A helyzetet nehezíti, hogy számos eltérő, nagy léptekkel változó technológia van használatban. Ide tartoznak a különböző hosszú távú ütemezők: a PBS, az LSF és a Condor [5] is.

Igaz ugyan, hogy a grid fejlesztés végső célja egy szabványosított módon elérhető világméretű szolgáltatás kialakítása, ennek megvalósulásáig azonban várhatóan még sok évet kell várni. Jelenleg az egyes grid rendszerek különböző, egymással inkompatibilis köztesréteg-szoftvert használnak. Mindezeket a technológiákat megismerni, használatukat megtanulni ahhoz, hogy egy általános problémát, például paraméterelemzést megoldhassunk a segítségükkel, felesleges energiákat von el a kutatás igazi feladataitól. Gyakori eset továbbá, hogy egy már megírt programot gyökeres átírás nélkül szeretnének használni a háttérben levő infrastruktúra változása, fejlődése után is. Ilyen változás lehet az, amikor egyetlen számítógépről áttérnek egy helyi fürt használatára, vagy pedig a fürtről térnek át valamely griden történő futtatásra.

3.2 Megoldási javaslat

Ezen nehézségek áthidalására hivatott a Saleve rendszer, egy nyílt forráskódú segédeszköz a C nyelvű, párhuzamos futásra is képes programok fejlesztésére. A Saleve egyaránt használható új programok készítésére és már meglevő, de egyszerre csak egy processzoron futni képes kódok gyors átírására. Az alapvető cél elfedni a háttérben megbújó különféle számítási technológiákat és egy ezektől független, könnyen elsajátítható metodológiát nyújtani olyan paraméterelemző alkalmazások gyártására, amelyek az egyszerű szekvenciális futáson kívül képesek a párhuzamos rendszereket is kihasználni.

1. ábra A Saleve használati lehetőségei



4. A Saleve működése

4.1. Kliens-szerver architektúra

A Saleve rendszer működésének megértéséhez tekintsünk egy C nyelven megírt, szekvenciális parameter study (PS) alkalmazást. A felhasználó számára az egyetlen újdonság az, hogy kis mértékben átalakítva az eredeti alkalmazását, el kell készítenie a Saleve klienst. A szükséges változtatás mindössze abból áll, hogy szét kell választani a programban a paramétertartomány felbontását, a részeredmények kiszámítását és a részeredmények összegzését, majd fordításkor össze kell szerkeszteni a Saleve fejlesztői könyvtárral.

A Saleve kliens futtatásakor kiszámítja az összes résztartományhoz tartozó részeredményt és összegzi azokat. Alapértelmezésben – az eredeti alkalmazáshoz hasonlóan – sorban egymás után indítja a részfeladatokat, de lehetőség van arra is, hogy párhuzamosan több részfeladatot indítson. Azonban a kliens legfontosabb képessége az, hogy el tudja küldeni saját bináris kódját és a bemeneti állományokat egy megadott Saleve szervernek.

Miután a kliens kérése, vagyis a program és az adatfájlok megérkeznek a Saleve szerverhez, a szerver minden résztartományhoz egy külön folyamatot indít el, amelyeket vagy helyileg hajt végre, vagy továbbküld egy gridbe vagy egy fürtbe a felhasználó számára tökéletesen transzparens módon *(1. ábra)*.

A szerver gondoskodik a továbbküldött feladatok felügyeletéről, hiba esetén újraküldésről és a részeredmények ideiglenes tárolásáról. Ebben a fázisban kliens a szerverrel való kapcsolatot megszakíthatja és később, akár egy másik kliens példány az azonosítást követően újra felépítheti.

A szerver a kiszámított részeredményeket folyamatosan visszajuttatja a vele kapcsolatban álló kliensnek. Amikor minden részeredmény megérkezett a klienshez, akkor a felhasználó által megadott módon kiszámításra kerül a végeredmény. A feladat végrehajtásának folyamatát a *2. ábra* foglalja össze.

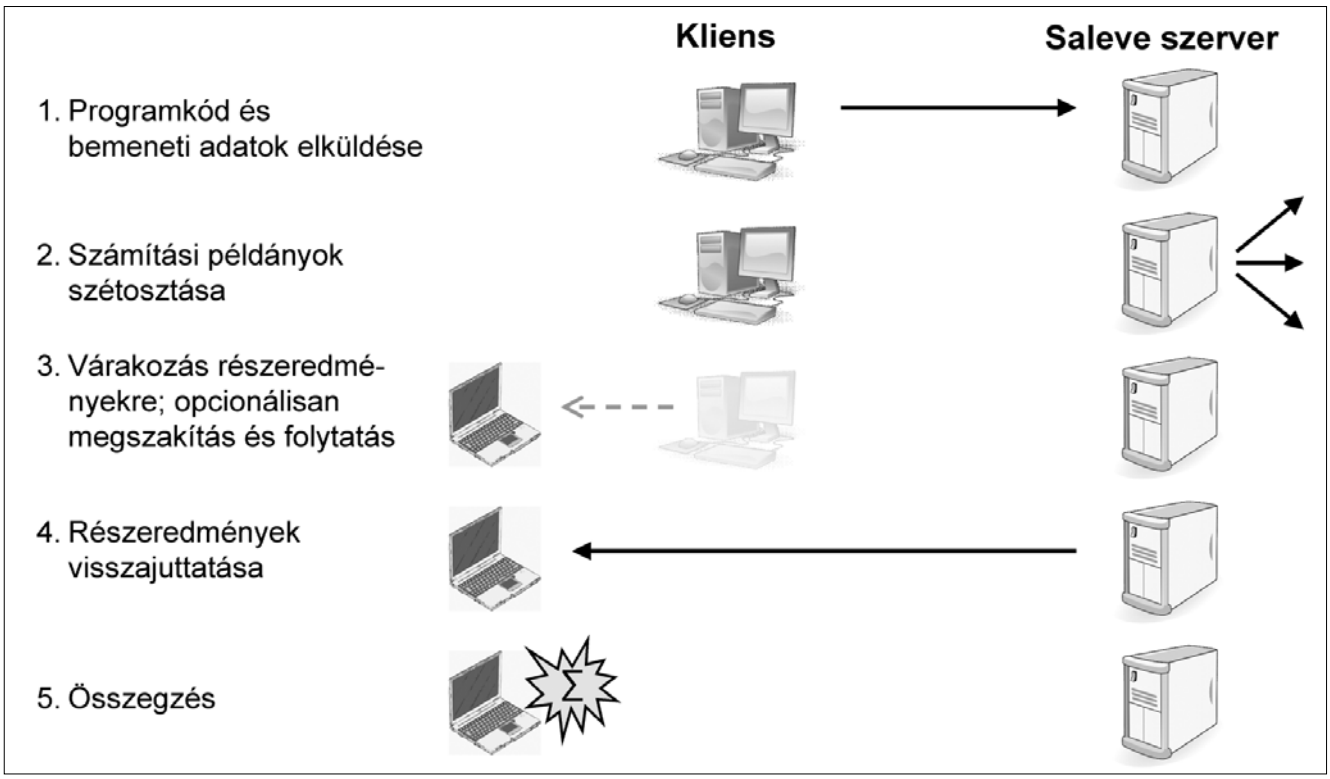
4.2. A szerver felépítése

A szerverrel szemben támasztott legfőbb követelményünk, hogy minél több elterjedt elosztott számítási környezetet támogasson, és emellett könnyen adaptálható legyen egy új ütemezőhöz vagy grides köztesréteghez. E szemléletet tükrözi a szerver komponenseinek két csoportra való felosztása: egyik típusú komponensek az általános, számítási környezettől független feladatokat végzik, a másik csoportot a plugin komponensek (kiterjesztések) képezik.

Az általános csoportba tartozik a kliensekkel való SOAP alapú kommunikációt végző komponens. A szerver tehát webszolgáltatásokat nyújt a kliensek felé, ezeken keresztül történik a feladat és a paraméterek feltöltése a szerverhez és a részeredmények visszajuttatása a klienshez. A webszolgáltatások a gSoap [6] implementációját használják. Az általános csoportba sorolandó továbbá a felhasználókat kezelő és a feladatkezelő komponens, melyeket részletesen [7,8] ismertet.

A kiterjesztések csoportja, a pluginok teszik lehetővé, hogy a szerver többféle, eltérő környezetekkel lehessen kapcsolatban, de egy új környezethez történő illesztés során ne legyen szükség a működés újratervezésé-

2. ábra A feladat végrehajtása



re. Minden egyes elosztott környezethez fejleszteni kell egy plugint, amely a környezetspecifikus kommunikációt kezeli *(3. ábra)*.

Eddig az alábbi környezetekhez készült Saleve plugin:

– a Saleve szervert befogadó gépen történő végrehajtás, ami a szervergépen párhuzamosan indítja el a feladatokat,
– Condor ütemezőnek [5] történő feladatbeküldés, mely a fürtöknél gyakran használatos feladatütemező,
– az EGEE grid infrastruktúrába továbbítás a gLite köztesrétegen keresztül.

4.3. Saleve és az EGEE infrastruktúra

A Saleve rendszer az előző szakaszban említett pluginnal támogatja az EGEE gridbe történő feladatbeküldést is. Egy Saleve plugin elkészítése elsősorban a megfelelő köztesréteg vagy ütemező interfészének ismeretét követeli meg és nem igényel nagy felkészültséget a Saleve architektúrájának terén. A plugin fejlesztésének folyamata egy absztrakt interfész-osztály implementálásából áll, ahol az adatkezelést segíti a Saleve programkönyvtár. A kihívást inkább a grid felé történő hitelesítésben és a grides feladatok menedzselésében találtuk. A feladatok gondozását a futtatást végző infrastruktúra megbízhatatlansága teszi szükségessé: bizonyos részfeladatok végrehajtása meghiúsulhat, ezeket újra be kell küldeni.

A gLite köztesréteg, az EGEE infrastruktúra szoftvermotorja olyan tanúsítványalapú hitelesítést és erőforrás-kiosztást alkalmaz, amelyben a felhasználókat és az erőforrásokat virtuális szervezetekbe (VO) csoportosítja. Ha egy felhasználó el kíván érni egy erőforrást, például feladatot szeretne beküldeni, akkor a tanúsítványából egy rövidlejáratú, úgynevezett proxy tanúsítványt kell generálnia, ezt csatolnia kell a beküldött feladathoz és rendszeresen megújítania. Ez az eljárás segít megóvni a hosszú távú tanúsítványt abban az esetben, ha a proxy tanúsítvány kompromittálódna.

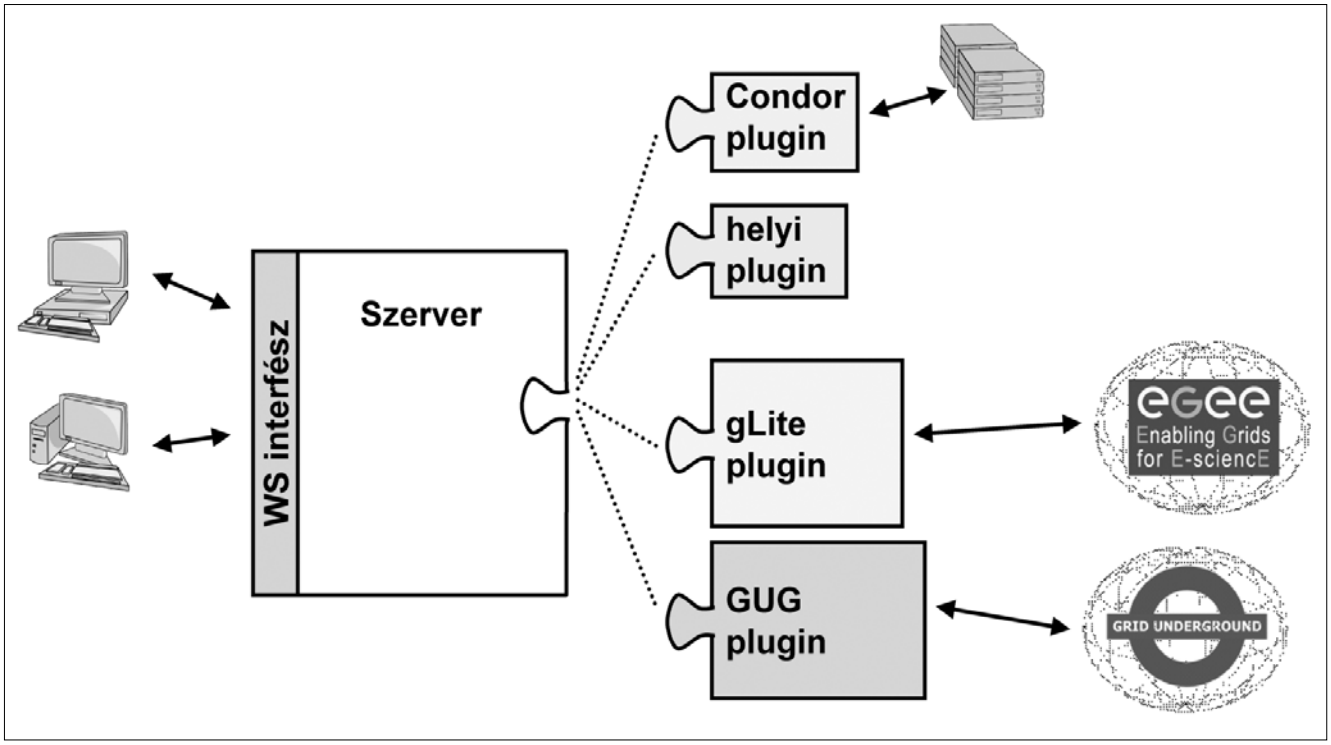
Jelenlegi megoldásunkban a Saleve szerver saját tanúsítványt tart birtokában, vagyis közvetlenül hozzáfér a grides erőforrásokhoz, hiszen tagja valamelyik virtuális szervezetnek. Így a proxy generálása és periodikus megújítása teljesen elrejthető a felhasználó elől, aki így nem észleli, hogy a feladata az EGEE gridben, vagy egy helyi Condor-fürtön futott-e le.

5. Összegfoglalás

A bemutatott Saleve rendszer átlátszó absztrakciós réteget képezve a különféle elosztott környezetek köztesrétege és ütemezője felett megkönnyíti a parameter study típusú párhuzamos alkalmazások fejlesztését. Legfőbb előnye, hogy az eredeti alkalmazás enyhén módosított példánya, a Saleve kliens többféle környezetben vagy akár a helyi gépen is futtatható változtatás nélkül, így technikai részletek ismerete nélkül is könnyen fejleszthetünk alkalmazásokat akár Európa legnagyobb infrastruktúrájába: az EGEE gridbe.

A továbbfejlesztési lehetőségek közül a közeli jövőben hangsúlyt helyezünk a pluginok dinamikus cseréjének fejlesztésére és a gridbe küldött feladatok jobb menedzselésére. Terveink között szerepel a kliens-szerver kommunikáció rugalmasabbá tétele webstream-ek segítségével. A Saleve projekt jelenlegi állapotáról [9] weblap ad tájékoztatást.

3. ábra A Saleve architektúrája



Köszönetnyilvánítás

E munka részben a Nemzeti Kutatási és Technológiai Hivatal Pázmány Péter programjának (RET-06/2005) támogatásával jött létre. A szerzők szeretnék kifejezni a köszönetüket az Európai Unió által támogatott EGEE projektnek (EU INFSORI-031688), valamint az NKFP MEGA (2_009_04) projektnek.

Irodalom

[1] D. Pasztuhov, A. Sipos, I. Szeberényi: Calculating Spatial Deformations of Reinforced Concrete Bars Using Grid Systems, MIPRO 2007 – Hypermedia and Grid Systems, Opatija, Croatia, 2007., pp.189–194.

[2] P. E. Black: Algorithms and Theory of Computation Handbook, CRC Press LLC, U.S. National Institute of Standards and Technology, 1999.

[3] I. Foster, C. Kesselman: The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers Inc., 2003.

[4] EGEE-II Information Sheet, 2007. http://www.eu-egee.org/sheets/uk/egee-ii.pdf

[5] D. Thain, T. Tannenbaum, M. Livny: Distributed Computing in Practice: the Condor Experience, Concurrency and Computation: Practice and Experience, 2005., pp.323–356.

[6] R. A. van Engelen, K. Gallivan: The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks, In Proc. of the 2nd IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002), Berlin, Germany, 2002., pp.128–135.

[7] Zs. Molnár, I. Szeberényi: Saleve: simple web-services based environment for parameter study applications. In Proc. of the 6th IEEE/ACM International Workshop on Grid Computing, 2005., pp.292–295.

[8] P. Dóbé, R. Kápolnai, Szeberényi: Simple grid access for parameter study applications. In 6th International Conference on Large-Scale Scientific Computations, Sozopol, 2007. (nyomtatás alatt)

[9] Saleve projekt, http://egee.ik.bme.hu/saleve/

[10] P. Kacsuk, Z. Farkas, G. Hermann: Workflow-level parameter study support for production Grids, In Proc. of the ICCSA'2007, Kuala Lumpur, 2007. Springer LNCS 4707, pp.872–885.

[11] P. Kacsuk, G. Dózsa, J. Kovács, R. Lovas, N. Podhorszki, Z. Balaton, G. Gombás: P-GRADE: A Grid Programing Environment. Journal of Grid Computing, Vol. 1, No.2, 2004. pp.171–197.

[12] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto, Jr., H.-L. Truong: ASKALON: a tool set for cluster and Grid computing, Concurrency and Computation: Practice and Experience, Vol. 17, 2005., pp.143–169.

Hírek

A biometrikus azonosító eszközök – ujjlenyomat-, íriszleolvasók, arcfelismerők – piaca jelentős fejlődésnek indult az utóbbi időben, Magyarországon is egyre több cég használ ilyen eszközöket. **FaceReading** elnevezéssel figyelemre méltó hazai fejlesztésű képfelismerő programot mutattak be december elején, amely képes arra, hogy akár egy egyszerű webkamerával, valós időben készített képeken is 95%-os pontossággal azonosítsa azokat a kulcspontokat, melyek segítségével az arc elemezhetővé válik és másodpercek alatt megállapíthatóak az illető alapvető személyiségjegyei, melyekből számos tulajdonságára lehet következtetni.

A FaceReading Kft. az arcelemzés sok évszázados tudományát ötvözte a mesterséges intelligenciára épülő technológiával, melynek nyomán az Arcolvasó neurális hálózatokból felépített algoritmusai a relatíve rossz minőségű, alacsony felbontású mozgóképen is képesek az arc kulcs-pontjait követni és azonosítani. A cégvezetők szerint a technológia forradalmasíthatja a munkaerő-kiválasztást, illetve annak hatékonyságát. A kulcspontok adataiból Nagy Judit ügyvezető, a cég humánerőforrás menedzsmenttel foglalkozó pszichológusa szerint olyan személyiségjegyekre lehet következtetni, mely egy adott munkakörre vonatkozó alkalmassági vizsgálatban is segíteni tudja a HR-esek munkáját.

Galambos József cégvezető a sajtótájékoztatón elmondta, hogy az alacsony hardverigényű, költséghatékony rendszer további alkalmazási területekkel is rendelkezik, ugyanis megfelelő "tanítás" után képes beazonosítani az arcon kívül más tárgyakat is, így például a járművek rendszámán túl felismeri karosszériájukat, vagy akár az azokon végzett módosításokat is. A technológia természetesen felhasználható az ellenőrzött be- és kiléptetés meggyorsítására, célszemélyek kiszűrésére civil és speciális területeken egyaránt.

Az Arcolvasó iránt főként az algoritmusok skálázhatósága miatt komoly a nemzetközi érdeklődés, a cég szakemberei egyelőre csak annyit árultak el, hogy jelenleg is több új alkalmazást fejlesztenek.

Grid technológia az allergiásokért

ÁDÁM RITA, BENCSIK ATTILA

Glia Számítástechnikai és Tanácsadó Kft.
{radam, abencsik}@glia.hu

Lektorált

Kulcsszavak: pollen, meteorológia, pollen-előrejelzés, meteorológiai adat, időjárás-előrejelzés, wise mind grid

A számítógépek kihasználatlanul hagyott kapacitását az időjárási viszonyok és a pollenkoncentráció kapcsolatának feltárására, az időjárás-előrejelzésen alapuló, minél pontosabb pollen-előrejelző rendszer kialakítására használjuk. A statisztikai összefüggések keresésekor nagy mennyiségű, több éves meteorológiai és pollen adatsorral dolgozunk, melyek közti bonyolult kapcsolatot adatbányászati módszerekkel kívánjuk felderíteni.

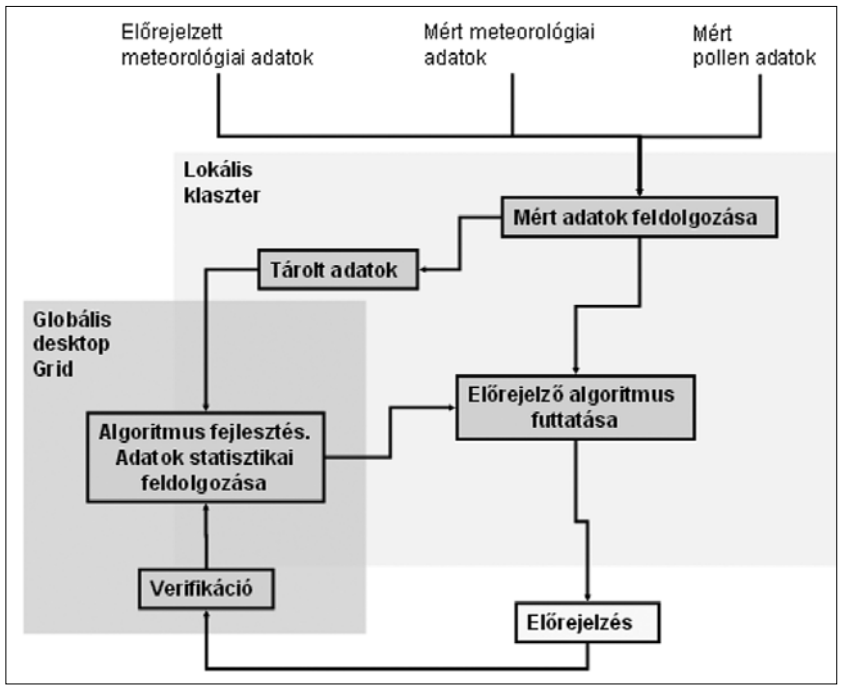
1. Bevezetés

Számos olyan tudományos és műszaki probléma létezik, amelynek megoldására nem lehet a szokásos modellezési eljárásokkal választ találni a rendszer bonyolultsága miatt. A statisztikai analízis és a nagy számítási teljesítmény kombinációja viszont lecsökkentheti a kutatás időtartamát. A masszív kombinatorikának nevezett kutatási módszer használatával az elméletekből származó feltételezések kísérleti tesztelése nagyságrendekkel lerövidülhet. Az eljárás alkalmazása a szokásos számítógépes kapacitás mellett megoldhatatlan feladat. Ezt a problémakört igyekszik áthidalni a GRID technológia. A projektben az asztali számítógépek kihasználatlanul hagyott kapacitását az időjárási viszonyok és a pollenkoncentráció kapcsolatának feltárására, az időjárás-előrejelzésen alapuló, minél pontosabb pollen-előrejelző rendszer kialakítására használjuk. A statisztikai összefüggések keresésekor nagy mennyiségű, több éves meteorológiai és pollen adatsorral dolgozunk, melyek közti bonyolult kapcsolatot adatbányászati módszerekkel kívánjuk felderíteni. Az általunk kidolgozott módszer a GRID technológia új alkamazását, az allergiás megbetegedések megelőzését tűzte ki célul. Ezt a procedúrát ismertetjük részletesen az alábbiakban.

2. A kutatás célja

A jelen cikkben ismertetett algoritmuskeresés során a meteorológiai változók (hőmérséklet, csapadék, relatív nedvesség, légnyomás, szélsebesség) és a levegőben jelenlévő pollenkoncentráció közötti kapcsolatot keressük. A minél pontosabb összefüggések feltárásának célja az időjárás-előrejelzésen alapuló pollen-előrejelző rendszer kialakítása. Az algoritmus-keresés során külön vizsgáljuk az egyes meteorológiai változók hatását a pollenkoncentráció alakulására, valamint ezeket súlyozzuk különböző időtávokkal. A meteorológiai viszonyok a pollent kibocsájtó növény fejlődését befolyásolják, így a későbbi pollenkibocsátó képességére is hatással vannak. Természetesen más-más módon befolyásolhatja például a parlagfüvet a fejlődési fázisban kapott napfénytartam, az akkor fújó szél vagy a lehulló csapadék. Ezért az egyes változókról már kiinduláskor feltételezzük, hogy a múltra vonatkozóan különböző súlyt kapnak. A célunk megtalálni azokat az algoritmusokat, amelyek a meteorológiai körülmények figyelembe vételével (legyen szó a mai napról, vagy akár az elmúlt fél évről), a meteorológiai előrejelzésen alapulva minél pontosabban megbecslik a várható pollenkoncentrációt. Az eredményeinket a kutatás lezárultával egy új pollen-előrejelző rendszer formájában tervezzük hasznosítani.

1. ábra Az előrejelző rendszer összetevői



3. Econet Wise Mind Grid

A feladat megoldására az econet által kifejlesztett Wise Mind Desktop Grid rendszert választottuk. A Wise Mind egy ipari felhasználásra kifejlesztett magas biztonsági paraméterekkel rendelkező Desktop Grid rendszer. A WM Desktop Grid rendszert kialakításából fakadóan lehet lokális és globális szinten is alkalmazni.

A lokális szint azt jelenti, hogy a teljes Desktop Grid „házon" belül ugyanazon tűzfal mögött van, azaz lokális hálózattal vannak a Desktop Grid erőforrásai összekötve. A globális szint azt jelenti, hogy az erőforrások interneten keresztül kapcsolódnak egymáshoz. A kutatási programban az algoritmuskeresésre a rendszer globális kiépítését használjuk, a napi pollen előrejelzést pedig az econet irodai hálózatára tepített lokális rendszer segítségével végezzük.

4. A GRID szerepe

A statisztikai összefüggések keresésekor nagy mennyiségű, tíz évre vonatkozó meteorológiai és pollen mérési adatbázist dolgozunk fel. Az algoritmus keresésekor a számítási hatékonyság növelésére alkalmazzuk a GRID technológiát. A GRID rendszerek alkalmasak hatalmas mennyiségű adat tárolására, illetve feldolgozására. Mi ebből a feldolgozás részt használjuk ki.

A projekt megvalósításához egy weboldalt hozunk létre, ahol a leendő résztvevők regisztrálják magukat, ezután letölthetik a kapcsolódáshoz szükséges programot és a feldolgozandó adatbázisokat. Ezután az internetre csatlakozott számítógépek felhasználatlan kapacitását hasznosítjuk és algoritmusokat tesztelünk rajtuk. Ezek az algoritmusok keresik a meteorológiai elemek és a levegőben mérhető pollenkoncentráció közötti kapcsolatot.

Első közelítésben lineáris kapcsolatot feltételeztünk a változók között:

$$\sum a_i*t+b_i*u+c_i*f... = \text{pollen koncentráció},$$

ahol
t – a hőmérséklet,
u – a szélsebesség,
így vettük figyelembe a többi változót is
(relatív nedvesség, légnyomás, csapadék)
a,b,c,... stb. – a keresett együtthatók
(lehetnek függvények is),
i – az időlépcső
(ez a különböző változóknál lehet más és más).

Az általunk előállított becslés és a valódi, mért pollenkoncentráció közti eltérés mérésére, azaz a módszer hibájára az átlagos négyzetes hibát választottuk több szakirodalmi hivatkozás alapján:

$$RMS = 1/n(\sum(m_i-e_i)^2)^{1/2}$$

ahol m_i – az i-edik mérés,
e_i – az i-edik előrejelzés,
n – az adatok száma.

Ezzel a hibafüggvénnyel mértük a későbbi algoritmusaink hibáját is.

Az első algoritmus keresés során kiderült, hogy lineáris kapcsolatnál bonyolultabb összefüggést kell keresnünk a koncentráció és a meteorológiai elemek között. Második közelítésben olyan függvényeket tartalmazó nemlineáris kapcsolatokat vizsgáltunk, amelyek a változók közötti bonyoult kapcsolatot feltételezéseink szerint reprezentálhatják.

Ilyen függvényekre példa:

$const_1*(\ln(x))^{const2}$
$\ln(const_1+cons_2*x+const_3*x^{const3})$
$(const_1*\exp(x))$
$const*\exp(x)$, $const_1*\exp(const_2*x)$

ahol
x – bármely változó, a konstans, pozitív egész szám.

A fenti függvények csak példák, hiszen külön algoritmusokat próbálunk ki mind az egyes meteorológiai elemekre, mind az év adott pollenterhelési időszakaira vonatkozóan.

Az algoritmusok kidolgozására egy vezérlő szerver osztja ki a feladatokat és értelmezi az egyes számítógépek eredményeit. A tesztek eredményét a résztvevők a weboldalon keresztül folyamatosan nyomon követhetik. A weboldal arra is szolgál, hogy a kutatás eredményeit folyamatosan publikáljuk, így az bárki számára hozzáférhető lesz.

A programban abban az esetben is részt lehet venni, ha nem rendelkezünk folyamatos internet kapcsolattal (ami manapság már viszonylag ritka). Ekkor az internetre való fellépéskor a GRID-hez szükséges kliens program feltölti az elvégzett számítások eredményeit a vezérlő szerverre. Állandó internetkapcsolat esetén az adatáramlás a vezérlő szerverre valódi időben megvalósul.

Miután a kapcsolatot feltártuk, a meteorológiai előrejelzésen alapuló pollen-előrejelzést tudunk készíteni oly módon, hogy az összefüggést alkalmazzuk a mért adatokra, emellett az időjárás-előrejelzést is figyelembe vesszük.

A fent leírt algoritmus-keresést először a parlagfűre, később a fennmaradó 34, hazánkban előforduló pollenre szeretnénk végrehajtani. Az így létrejövő rendszer a teljes országot, a teljes allergén időszakot és az összes hazai pollent lefedő előrejelzést szolgáltatna.

A projekt egy nagyobb lélegzetű hazai együttműködés, a 2005-ben elkezdődött HaGrid projekt [1] egyik részfeladataként fut. A HaGrid az econet.hu Informatikai Zrt, az Országos Meteorológiai Szolgálat, a MTA Számítástechnikai és Automatizálási Kutató Intézet, a Glia Számítástechnikai és Tanácsadó Kft., a DDC Automatika Kereskedelmi és Szolgáltató Kft. és az Env-in-Cent Környezetvédelmi Tanácsadó Iroda együttműködésében valósul meg. A program célja, hogy a Desktop GRID technológiát olyan formára dolgozza át, hogy az alkalmas legyen mind zárt, vállalati környezeten belüli, mind azon kívüli feladatok megoldására.

A megvalósításhoz a résztvevők három központot (SZTAKI, OMSZ, econet) hoznak létre, amely a rendszer

alapját alkotja. A GRID rendszert ezután különböző célú alkalmazásokkal tesztelik, ezek egyike a pollen-előrejelző rendszer. Ezzel a projekt egy jövőbeni, hazai szolgáltató rendszer alapjait igyekszik megteremteni, amelyben a PC tulajdonosok becsatlakozhatnak a (vállalati és egyetemi) kutatás-fejlesztésbe.

A kutatás témájához kapcsolódik még a SZTAKI, a BME, az ELTE és a Compaq együttműködésével létrejöttt SzuperGrid [2] projekt (2002-2003), amelynek keretében a SZTAKI adaptálta a P-GRADE programfejlesztő rendszert a gridhez [3]. Ennek segítségével intézményen belül a különböző, heterogén informatikai erőforrásokat összekapcsolták. A P-GRADE egyik jelentős alkalmazási területe az Országos Meteorológiai Szolgálatnál futtatott ultrarövid távú előrejelzés informatikai hátterének biztosítása volt.

A grid technológiát a meteorológia több területén alkalmazzák, ezek közül napjainkban kiemelkedően fontos a klímakutatás. A különböző meteorológiai-éghajlati modellek a legnagyobb kapacitású számítógépeket igénylik szerte a világon. A klímakutatásban világviszonylatban élen járó Hadley Centre [4], a brit meteorológiai szolgálat, klímakutató intézete több egyetemmel karöltve grid rendszerre alapozva a században várható éghajlatváltozásokat próbálja minél pontosabban előrejelezni [5].

5. Összefoglalás

A fent részletezett módszer a GRID rendszerek újfajta alkamazását mutatja be. A programhoz csatlakozó résztvevők számítógépein olyan algoritmusokat kívánunk futtatni, amelyek nagy méretű, több évre vonatkozó meteorológiai és pollen-adatbázisok feldolgozásával megkeresik a köztük fellelhető összefüggéseket és segítenek a pollenterhelés előrejelzésének megvalósításában.

Irodalom

[1] Hungarian Advanced Grid
http://hagrid.hu

[2] Magyar Szuperszámítógép Grid
http://mszgrid.iif.hu/

[3] Kacsuk, P.:
A magyar grid rendszerek és fejlesztési irányaik. VIII., Országos (Centenáriumi) Neumann Kongresszus, Budapest, 2003.

[4] Hadley Centre for Climate Prediction and Research
http://www.metoffice.gov.uk/research/hadleycentre/

[5] Climate prediction project
http://www.climateprediction.net/

Hírek

A **Cisco** újgenerációs Unified Wireless Network egységes, vezeték nélküli hálózati megoldásának része a Cisco Aironet 1250 sorozatú hozzáférési pont, a világ első vállalati szintű 11n szabványnak megfelelő készüléke, valamint a 48 Gbit sebességű, kiválóan méretezhető Cisco Catalyst 6500 alapú WLAN-vezérlőrendszer, illetve a Unified Wireless Network 4.2-es változatában debütáló vezetékes és vezeték nélküli szolgáltatások. A Catalyst kapcsolócsalád egyetlen Ethernet-portról is megoldja a két rádiósávos Aironet 1250 sorozatú hozzáférési pontok áramellátását.

A Cisco új Aironet 1250-je a világ első Wi-Fi tanúsítvánnyal rendelkező **802.11n 2.0**-s szabványtervezetét követő hozzáférési pontja, és az egyetlen olyan kereskedelmi forgalomban kapható termék, amely része volt a Wi-Fi Alliance 802.11n 2.0 tervezet tesztelési környezetének, amellyel a későbbiekben az összes egyéb terméket hitelesíteni fogják a Wi-Fi-együttműködés szempontjából. A Cisco a rádiófrekvenciás kommunikációban szerzett tapasztalatát felhasználva ötszörös teljesítménynövekedést ért el a többcsatornás bemenettel és kimenettel rendelkező, úgynevezett MIMO-technológia terén. Ezzel sokkal megbízhatóbbá tehető a vezeték nélküli lefedettség, ami különösen a rádiófrekvenciás felhasználás szempontjából problémás területeken jelent nagy előnyt, mint például az egészségügy, a felsőoktatás, a raktározás, a logisztika és a gyáripar.

A Cisco 802.11n technológiával felvértezett, újgenerációs vezeték nélküli termékei a következőket nyújtják:

– A Catalyst 6500-hoz tartozó, a 802.11n technológia fokozatos, vagy azonnali, nagy volumenű bevezetését egyaránt támogató Cisco WiSM szolgáltatásmodul révén a felhasználók további vezérlők hozzáadásával bővíthetik a kapacitást igényeik szerint. Rugalmas méretezhetőségével a teljes szervezetre kiterjedő WLAN-bevezetések is gyorsan megoldhatók.

– A Cisco az új Unified Wireless Network Software nevű programjának 4.2-es verziója kibővített mobil szolgáltatásokat biztosít. Idetartozik a vállalati "mesh" alapú WLAN-hálózat, a hozzáférési pontok figyelése és a migrációs segédletek, valamint a vendégfelhasználók számára biztosított egységes vezetékes és vezeték nélküli hozzáférés. Emellett ezek a szolgáltatások fejlett vezeték nélküli hangátviteli funkciókkal, a távoli elérési pontok hibatűrő, átfedő elhelyezésével és integrált spektrumanalízissel rendelkeznek.

– A továbbfejlesztett kliensprogramok, így a Cisco Secure Services Client 5.0 által is kínált új lehetőségek az egyszerűbb vállalati szintű, egyetlen kliensbiztonsági és felügyeleti keretrendszer kialakítását szolgálják.

Hungary in the EGEE project
grid, e-science, EGEE

The goal of EGEE2 (Enabling Grids for E-sciencE 2) project is to develop and create an international research grid. Hungary has been participating in that for years now by operating clusters – or so called sites – that are the basic cells of the infrastructure, and by developing user applications. This article tries to give a brief review of the project, its exact goals, its operational structure and about the details of the Hungarian participation.

SEE-GRID:
The South Eastern European grid infrastructure
Keywords: grid, SEE-GRID project, grid infrastructure

The Laboratory of Parallel and Distributed Systems of the Hungarian Academy of Science (MTA SZTAKI LPDS) is participating in the building up and maintenance of the South Eastern European (SEE) grid infrastructure. In the frame of the SEE-GRID project, the tight collaboration among the partners in the recent years produces constantly growing computation power and storage size within the grid infrastructure. The SEE grid infrastructure provides freely accessible services for grid users and application developers within the SEE region, and it has the aim to strengthen scientific collaboration and cooperation among participating SEE communities. In this paper we give an overview about the SEE-GRID (South Eastern European GRid-enabled eInfrastructure Development) projects, we introduce the established SEE grid infrastructure and provide information on how users or application developers can access the resources and give examples about the main research areas using the SEE grid infrastructure.

Security issues grids
Keywords: grid, data-security, break-in, incident, vulnerability, identity spoofing, x.509, cluster

Connecting computing clusters together into an international network with proper broker and manager layers seems to be an easy recipe to create a grid-system. However, while the local clusters are relatively small therefore easily protectable and usually even world-wide-web separated, the grid-systems are usually connected via public internet connections. That arises lots of serious security-related questions. The article tries to introduce these questions and the possible solutions.

Task scheduling in desktop grids
Keywords: grid, task scheduling, scalability, hierarchical desktop grids

In the paper we present the task scheduling questions related to a relatively new grid trend, the desktop grids. In spite of the serivce-based grid systems, in case of desktop grids the user tasks are placed on a central server and donors offering their free CPU cycles download them from the server, and upload results after processing. So we do not search an adequate resource for the task but donors query tasks from a central repository. In the paper we introduce desktop grids, show a few methods related to their scalability, intorduce the hierarchical desktop grid concept, task scheduling questions and possible related algorithms.

The evolution of Grid Brokers:
union for interoperability
Keywords: Grid Brokers, grid interoperability, meta-brokering, matchmaking

Grid resource management is probably the research field most affected by user demands. Though well-designed, evaluated and widely used resource brokers, meta-schedulers have been developed, new capabilities are required, such as agreement and interoperability support. Existing solutions cannot cross the border of current middleware systems that are lacking the support of these requirements. In this paper we examine and compare different research directions followed by researchers in the field of Grid Resource Management, in order to establish grid interoperability. We propose a meta-brokering approach, which means a higher level resource management by enabling communication among existing Grid Brokers and utilizing them.

Using grid systems
in designing reinforced concrete beams
Keywords: grid, portal, industrial application, reinforced concrete beams, parallel computing

In our paper we present a real, engineering problem, its solution with a very efficient parallel algorithm, and a web tool which makes development of user interfaces easier. Using the tool the users can submit jobs easily in parallel and grid jobs, and start command-line applications.

Saleve:
toolkit for developing parallel grid applications
Keywords: grid application development, parameter study

We present Saleve, a developers' tool to aid the creation of parallel running parameter study (PS) applications. It is prepared to cooperate with several distinct parallel computing systems, and grid middleware systems. Without the need of knowing technical details of any specific middleware, Saleve enables for researchers to develop new parallel programs easily.

Development of pollen information system
using grid technology
Keywords: grid, meteorological data, pollen forecast, weather forecast, wise mind grid

We are using the ineffective resources of computers for discovering the relation of weather conditions and pollen concentration. Statistics based on databases of meteorological and pollen data covering several years which we use for searching complex connections by data mining.

Summaries • of the papers published in this issue