Munkában a böngészők

Molnár Gábor, Schnell Henrik, Szarvas Attila, Szeberényi Imre
BME IK, BME IIT
{schnell.henrik, molnar.gabor, szarvas.attila, szebi} @iit.bme.hu

Bevezetés

A kutatási feladatok egy része jelentős számítási kapacitást igényel. Ezeket a feladatokat többnyire szuperszámítógépek és/vagy elosztott számítási rendszerek segítségével oldják meg. Az elosztott rendszerekben a különálló számítógépek és számítógép fürtök elosztva dolgoznak a számukra kiosztott feladatokat, majd egy kommunikációs csatornán továbbítják az eredményeket a központi gépre. Itt a beérkezett adatok alapján létrehozhatók újabb feladatok, illetve elvégezhető az eredmények összesítése, értékelése.

A megoldás előnye a szuperszámítógépekkel szemben a rugalmasság, a skálázhatóság és a költséghatékonyság. Segítségével lehetővé válik, hogy sok, egyenként kisebb teljesítményű egység révén összességében hatalmas számítási teljesítményre tegyünk szert. Az elosztott számítások egy gyakran alkalmazott architektúrája a *grid*, amely különálló számítógépek erőforrásait szervezi egy egységes rendszerbe, lehetővé téve, hogy a résztvevők valamennyi állomás egységesített teljesítményét vegyék igénybe feladataik futtatásához. A gridek létrehozói és felhasználói között megtaláljuk a világ számos kutatóintézetét és egyetemét. Az erőforrások közösítése két komoly előnnyel jár a számukra: egyrészt a nagy teljesítmény lehetővé teszi, hogy feladataikat sokkal rövidebb idő alatt végezzék el, másrészt a számos résztvevőnek köszönhetően eszközeik sohasem maradnak kihasználatlanul.

A nagyteljesítményű processzorok és grafikus kártyák hétköznapi felhasználók közti elterjedése révén egy új potenciális erőforrás jött létre, melynek kiaknázásához a szélessávú internethez való hozzáférés révén ma már minden technológiai feltétel adott. Már az 1990-es évek végén létrejöttek az első *közösségi számítási projektek*, amelyek önkéntes alapon szerveződtek és a felhasználók különböző kutatási célok érdekében kínálták/kínálják fel saját erőforrásaikat. Ma már kevésbé ismert, bár akkoriban igen nagy port kavart az 1997-ben indított *distributed.net* projekt [3], amivel 250 nap alatt sikerült egy 56 bites RSA kulcsot megtörni, ami igen jól bizonyította az önkéntesen összeadott erőforrások erejét. Ma már csupán érdekesség, hogy a Berkeley egyetem NOW (Network of Workstations) projektje [4] 97-ben 10 GFlops teljesítményt ért el, amivel az akkori top 500-as lista 190. helyére kerülhetett volna. A számos önkéntes projekt közül az egyik legismertebb a milliós felhasználói bázist maga mögött tudó, és még jelenleg is aktív *SETI@home* projekt [5], mely

a Berkley egyetemen kifejlesztett *BOINC* [6][7][8][9] kliensprogram révén fér hozzá a vállalkozó kedvű emberek számítógépeihez.

A kliensprogramnak kulcsszerepe van ezekben a vállalkozásokban, mivel ez teszi lehetővé a munkaállomások közötti kommunikációt és a feladatok végrehajtásának menedzselését. A felhasználók ennek segítségével válhatnak részesévé a közösségi projekteknek, a kutatók pedig hozzáférhetnek az erőforrásokhoz. A BOINC kliensszoftvere minden jelentősebb operációs rendszerhez elérhető, és közel 40 tudományos kutatáshoz csatlakozhatunk a segítségével.

Bár a felhasználói számok önmagukban tekintve impozánsak,[1] ha figyelembe vesszük a web 2.0-es robbanás óta aktívan internetező, jelenleg a különböző közösségi oldalak körül csoportosuló felhasználók tömegeit, látható, hogy ezek a projektek csupán egy viszonylagosan szűk, az adott kutatások iránt érdeklődő rétegeit tudják megszólítani. Ebben szerepet játszik a BOINC kliens telepítésének viszonylagos nehézsége. A telepítési folyamat során a kliens beépül az operációs rendszer felületébe, ütemezője alapértelmezés szerint automatikusan indul és a háttérben folyamatosan fut. A kliens számos a számítógép felhasználásához tartozó beállítást tartalmaz, mely egyeseket elriaszthat. A projekteket a felhasználónak egyenként ki kell választania és regisztrálnia kell a hozzájuk való csatlakozáshoz.

A *Web2Grid* [1] projekt keretében létrehozott Web Computing megoldás minden eddiginél közelebb viszi a felhasználókhoz az elosztott számítások világát. Napjaink meghatározó informatikai tendenciája az alkalmazások webes változatainak megjelenése, melynek során a különböző operációs rendszereken működő, natív kódú programokat böngészőben futtatható változatok cserélik le. Ezeket a programokat nem kell telepíteni, és operációs rendszertől függetlenül elindíthatók egy támogatott böngésző segítségével. A kényelemnek és a webes közösséghez való közeliségnek köszönhetően ezek az alkalmazások sokkal gyorsabban terjedhetnek hagyományos társaiknál és rövid idő alatt óriási felhasználói táborra tehetnek szert.

A Web Computing [1] megoldás a webes alkalmazások valamennyi előnyét kiterjeszti az elosztott számítási feladatokra, ráadásul könnyű integrálhatóságával nemcsak önállóan, hanem bármely internetes felületen megjelenhet. A felhasználóknak nem kell az egyes projektek kiválasztásával foglalkozniuk, elegendő eldönteniük, hogy hozzájárulnak-e számítógépeik közösségi számítási projektekben való felhasználásához.

[1] Egy adott időpontban nagyságrendileg 300 000 aktív felhasználó van jelen a rendszerben.

1. Célok

A megoldás középpontjában egy böngészőben futtatható, általános célú függvénykönyvtár létrehozása áll, mely lehetővé teszi a közösségi számításokban való webes részvételt. A könyvtárral olyan webes alkalmazásokat lehet létrehozni, amelyek az elosztott számításokban már széles körben alkalmazott és jól bevált BOINC szerverekkel tudnak kommunikálni. A könyvtár lehetővé teszi a jelenlegi BOINC kliens által betöltött valamennyi fontos funkció böngészőben történő megvalósítását.

A minél gördülékenyebb működés és a gyors elterjedés érdekében biztosítani kell, hogy a könyvtár működéséhez ne kelljen semmiféle kiegészítőt telepíteni, egy modern böngésző önmagában is képes legyen a futtatására. A szerverekkel való kommunikáción túl a könyvtár azt is lehetővé teszi, hogy a tudományos számításokat is a böngésző motorja végezze el kliensoldali szkriptek végrehajtásával anélkül, hogy a felhasználói élményt rontaná.

Azáltal, hogy a folyamat egészét a böngészőben tartjuk, két fronton is elősegítjük a közösségi számítások elterjedését. Egyrészt a böngésző által értelmezett parancsfájlok az operációs rendszertől többi részétől elszigetelve futnak. A jelenlegi gyakorlattal ellentétben, ahol az operációs rendszeren futó natív kódot alkalmazunk, ez komoly biztonsági előrelépés, amely segíthet megnyerni a felhasználók bizalmát. Bízva a böngészők biztonságosságában csökkenthetjük a futtatni kívánt kódok ellenőrzésére fordított erőfeszítéseket. A BOINC projektjeibe jelenleg csak szigorú ellenőrzési procedúrát követően kerülhetnek kódok. Ennek a folyamatnak az egyszerűsítése a számítási feladatok megalkotóira is bátorítóan hathat.

A függvénykönyvtár felhasználására számos lehetőség kínálkozik, melyek közül a fejlesztés során három jól megkülönböztethető területet tartunk szem előtt. A legkézenfekvőbb egy böngészőben futtatható, tudományos alkalmazás megvalósítása, amely mind megjelenésében, mind funkcióiban a jelenlegi BOINC kliensre hasonlítana. Legfőbb előnyét a telepítési folyamat elhagyása jelenti, amely egyrészt könnyebbé teszi új felhasználók bevonását, másrészt a már meglévő felhasználóknak lehetővé teszi, hogy egy böngésző segítségével bármilyen számítógépről bejelentkezzenek, és számításokat végezzenek. Az elképzelés a tudományos számítások iránt érdeklődő, részben jelenleg is aktív felhasználókat célozza meg, a webes kialakítás viszont az ő esetükben is előnyökkel jár. Egy-egy már aktív felhasználó új, natív klienssel nem rendelkező számítógépeket vonhat be a

számításokba, illetve meggyőzheti a telepítési procedúrától ódzkodó ismerőseit a csatlakozásról.

A második lehetséges alkalmazási terület a manapság nagy figyelemmel kísért közösségi oldalak összekapcsolása az elosztott számítási megoldásokkal. A kizárólag a böngészőben elérhető technológiákat felhasználó eljárások egyrészt lehetővé teszik a függvénykönyvtár funkcióinak integrálását egy a közösségi oldalakon futtatható alkalmazásba. Ezek az alkalmazások ma nagyon népszerűek, és ha képesek felkelteni néhány felhasználó figyelmét, akkor a környezetükből adódóan nagyon gyorsan terjednek. A Web Computing függvénykönyvtára a böngésző erőforrásait alkalmazza a számításokkal kapcsolatos teendőkhöz, nem tartalmaz azonban semmiféle megkötést a megjelenésre vonatkozóan. Tetszőleges, a felhasználók számára érdekes alkalmazás elképzelhető tehát, amely a közösségi számításokat népszerű köntösbe bújtatja.

Maguk a közösségi oldalak nemcsak teret, hanem munkát is adhatnak az elosztott számítási megoldásoknak. A multimédiás funkciók bővülésével egyre több számításigényes feladat jelent meg, melyeknek legnyilvánvalóbb példája a fényképek és videók feltöltése. A képek átméretezésére, a videók tömörítésére a központi erőforrások helyett közösségi számítási megoldásokat is lehetne alkalmazni.

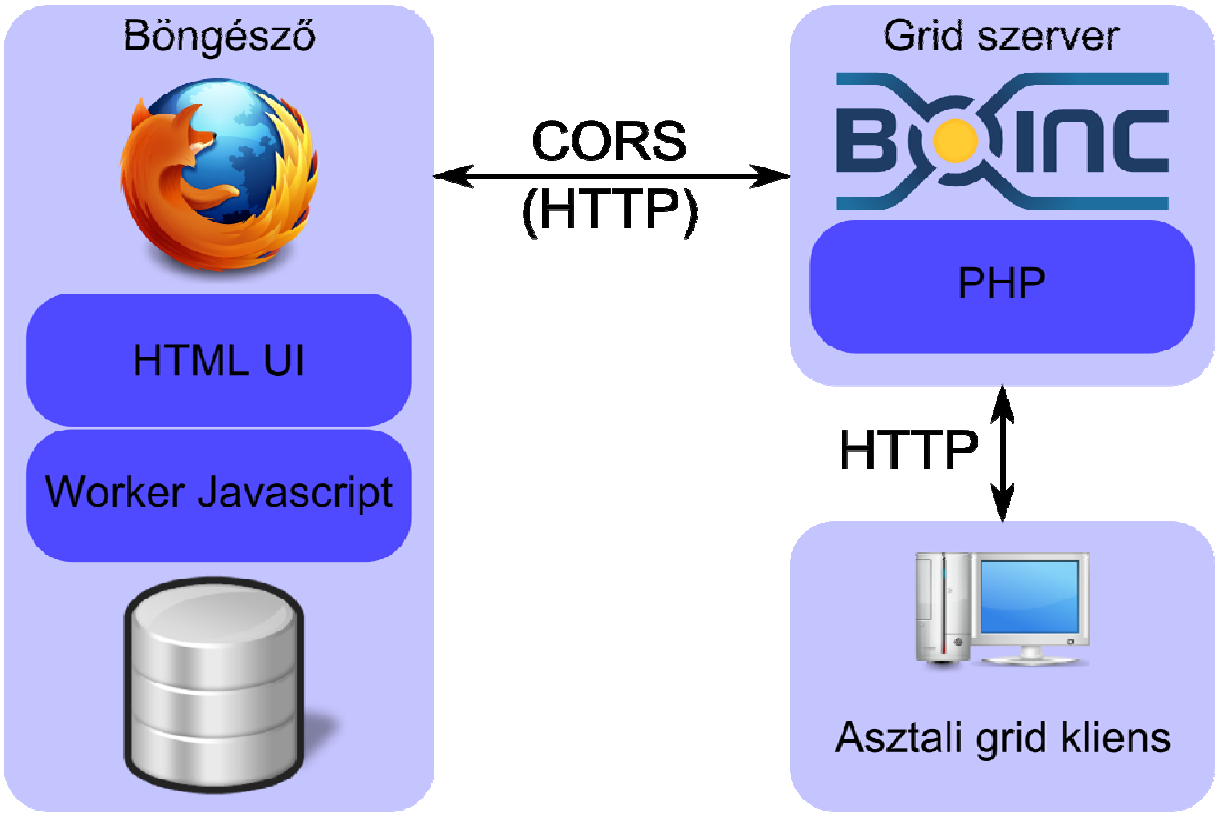
A harmadik, komoly haszonnal kecsegtető lehetőség az új generációs webes szolgáltatások és alkalmazások támogatása biztonságos, üzleti alapú grid platformmal. A könyvtár lehetővé teszi, hogy a számításokat tetszőleges profilú weboldalon vagy webes alkalmazás részeként a háttérben végezzük el. Különös gondot kell fordítani természetesen a felhasználók tájékoztatására, és fel kell ajánlani számukra a lehetőséget, hogy a számítási feladatokat szüneteltessék, vagy teljes mértékben letiltsák.

2. Architektúra

A Web Computingre épülő alkalmazások ugyanazokhoz a BOINC szerverekhez tudnak kapcsolódni, amelyek jelenleg is részt vesznek az elosztott számítási projektekben. A projektek működtetői igyekeznek minél több számítási teljesítményre szert tenni, ezért a számításokért felelős kódokat minden népszerű platformra lefordítják, és a szerver mindig a kliensnek megfelelő verziót küldi el a felhasználónak.

A különböző platformokon futó kódokat a BOINC egyenrangúan kezeli, valamennyi dolgozhat ugyanazon az adathalmazon. A Web Computing alkalmazásai saját platformnévvel rendelkeznek, melyet be kell jegyezni az adott BOINC szerver adatbázisába. A böngészőben futó kliensek számára a szerver a számítási program JavaScriptben [20] megírt változatát

fogja elküldeni. A webes alkalmazások ezért nem igényelnek új számítási infrastruktúrát, vagy új feladatokat, hanem a már meglévő rendszerhez kapcsolódnak.



A Web Computing architektúrája

A felhasználóknak csupán egy böngészőre van szükségük, amely szabványos technológiák felhasználásával kommunikál a grid szerverrel és futtatja a számításokat. Az új platform az asztali klienssel rendelkező felhasználók táborát egészíti ki többlet erőforrásokat bevonva.

3. Technológia

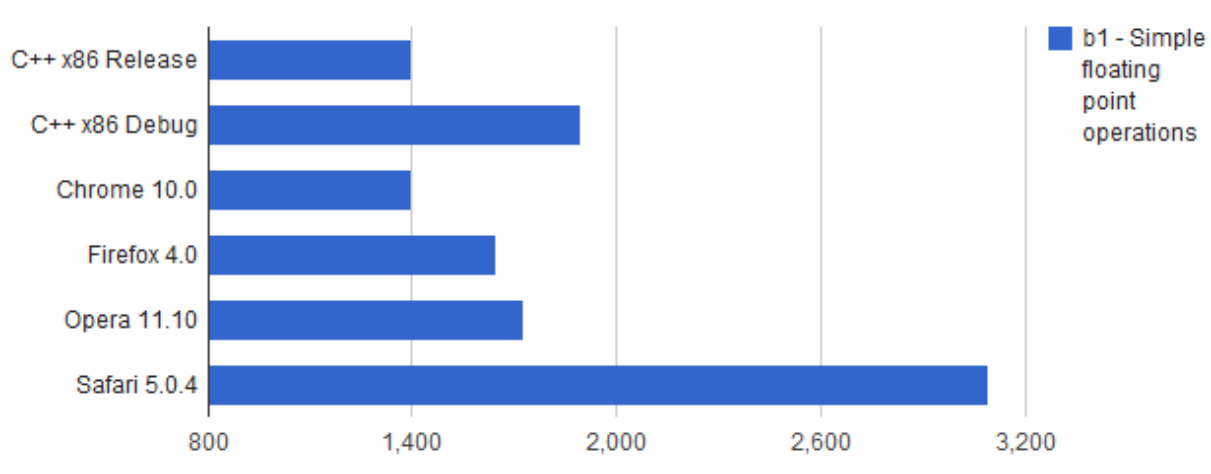
A projekt által kitűzött célok megvalósítása néhány éve még komoly akadályokba ütközött volna. A BOINC szerverekkel való kommunikációt, a komplex funkciókkal rendelkező alkalmazások kialakítását és a komoly számítások zökkenőmentes futtatását a böngészők végleges változataiban most debütáló *HTML5* [10] szabványcsoport és a *Web Workers* szabvány teszi lehetővé.

3.1. A fejlesztés nyelve

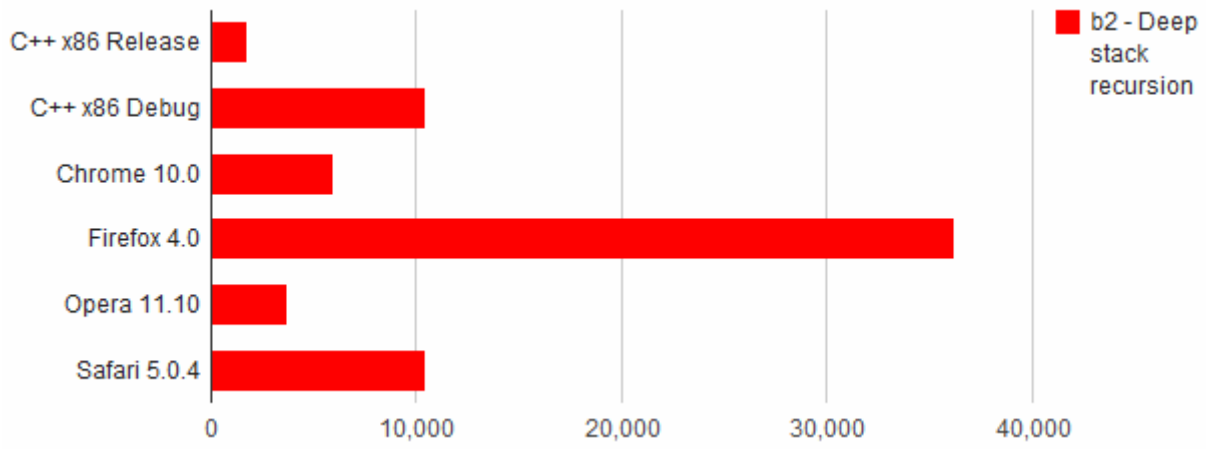
A fejlesztés során az egyik legfontosabb kritérium az volt, hogy a könyvtár kizárólag a böngészők által rendelkezésünkre bocsátott eszközöket használja fel, ne legyen szükség további kiegészítők telepítésére. Ebből adódóan egyedül a *JavaScript* nyelv jöhetett szóba. A

JavaScriptet régóta alkalmazzák interaktív webes elemek megvalósításához, a manapság népszerű szolgáltatások mindegyike erőteljesen épít a használatára. Megoldásunk ugyanakkor újszerű kihívás elé állítja a böngészőket. Mivel a keretrendszeren túl a tudományos számításokat végző programok is ezen a nyelven készülnek, az esetünkben huzamosabb ideig futó, a processzort teljes mértékben kihasználó szkriptekről van szó.

A feladat két aspektusát kellett megvizsgálni. Az egyik arra vonatkozik, hogy a JavaScript kód, illetve az egyes böngészők alkalmasak-e egyáltalán számításintenzív szkriptek futtatására. A tudományos számítások jellemzően kétfajta erőforrást vesznek igénybe. A fizikai szimulációkkal kapcsolatos számítások a lebegőpontos műveletvégzési sebességre, a bonyolult programok a stack kezelésének és a függvényhívásoknak a sebességére érzékenyek. Összeállítottunk két tesztet, amelyek az említett műveletek intenzív igénybevételével terhelték a rendszert, és megmértük, hogy a feladatok különböző nyelveken implementálva, illetve különböző JavaScript implementációkban mennyi ideig futnak.



1. teszt Egyszerű lebegőpontos műveletek



2. teszt Mély rekurzió

A mérések során biztató, sőt egészen meglepő eredményeket tapasztaltunk. *A legújabb böngészők JavaScript-végrehajtási sebessége az általunk vizsgált feladatok esetén a natív kódéval vetekszik.* Az utóbbi időben szinte valamennyi böngésző új verzióját a JavaScriptek végrehajtásának jelentős gyorsításával hirdették, amelyet az *AJAX* alkalmazások széleskörű elterjedése kívánt meg. Nem vitás, hogy a böngészők nagyon komoly előrelépést tettek ezen a téren, a sebesség tekintetében egyértelműen alkalmassá váltak a komoly teljesítményt igénylő számítások elvégzésére is.

Az intenzív számítások végzésére vonatkozó másik kérdés a felhasználói élmény garantálásával volt kapcsolatos. Egészen a közelmúltig a böngészők egy szálon hajtották végre kódjukat, és ugyanezen a szálon futtatták a szkripteket is. Ez ahhoz vezetett, hogy amennyiben egy szerencsétlenül megírt szkript egy tömbben nagy mennyiségű számítást tartalmazott, a megjelenített honlap kezelőeszközei, sőt rosszabb esetben magának a böngészőnek a felülete is „megfagyott", a felhasználó parancsaira érzéketlenné vált. Erre a problémára a *Web Workers* [16] szabvány nyújt megoldást.

A szabványban leírt *worker*-ek külön szálon végrehajtott, a böngészőben futó fő programtól elszigetelt objektumok, amelyek JavaScript kód futtatását teszik lehetővé a „háttérben". A worker a honlap DOM-jához nem fér hozzá, a főablakbeli kóddal csak üzenetek formájában tud kommunikálni, amely a benne futó számításoknak már nemcsak az operációs rendszertől, de a böngésző más részeitől való teljes elválasztását is jelenti. Ennek köszönhetően a biztonság terén sem kell csupán a böngészőt fejlesztő csapat hozzáértésére hagyatkoznunk, hanem a worker-ek következetes alkalmazásával magunk is gondoskodhatunk arról, hogy a BOINC szerverről letöltött kód ne tehessen kárt a rendszer más részeiben.

A külön szálon való futásnak köszönhetően a böngésző és a főablakban futó felhasználói felület fennakadásmentesen működik. Többmagos processzorok esetén több worker indításával minden rendelkezésre álló erőforrást kihasználhatunk.

Ahogy az előbbi meggondolások mutatják a JavaScript tökéletesen alkalmas arra, hogy a böngésző elosztott számítási feladatokat futtasson. Nagyobb projektek esetén a fejlesztők számára ugyanakkor szegényesnek tűnhetnek a nyelv által felkínált szolgáltatások. Ezért a fejlesztéshez közvetlenül a *haXe* [11][12][16] nyelvet használjuk. A haXe egy nyílt forrású, aktív közösséggel rendelkező, típusos, objektumorientált nyelv, amely nagy tudású eszközökkel segíti a fejlesztőket. Bőséges szolgáltatásai révén a haXe-ben megírt projektek áttekinthetőbbek, nehezebb bennük hibázni és a későbbiek során is könnyebben

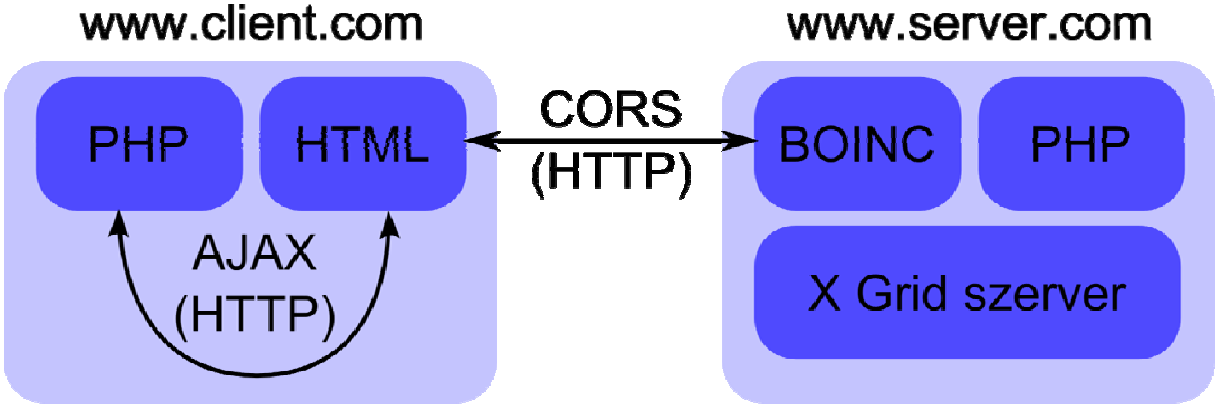
módosíthatóak. A haXe projekteket számos nyelvre le lehet fordítani, az esetünkben a JavaScript a célnyelv. A haXe-ből generált JavaScript kód jól optimalizált, és jól is olvasható.

3.2. A Web Computing és a BOINC kapcsolata

A Web Computingra épülő szolgáltatások megjelenhetnek egy tetszőleges honlapba beágyazottan, vagy önálló webes alkalmazásként. Mindkét esetben egy webes kliensről beszélhetünk, hiszen a könyvtár a BOINC kliensszoftver funkcióit látja el. A felhasználó böngészőjével meglátogatja a klienst elérhetővé tévő szervert, amely példánkban a *www.client.com*. Innen tölti le böngészője a Web Computingra épülő JavaScript alkalmazást.

A www.client.com és a felhasználó viszonylatában az alkalmazás úgy működik, mint bármely más AJAX [19] megoldás. A Web Computing alkalmazások rendelkeznek ugyanakkor egy fontos, és ritkán látott tulajdonsággal: a kliensoldali szkript képes más szerverekkel, esetünkben BOINC kiszolgálókkal is közvetlenül felvenni a kapcsolatot a www.client.com teljes megkerülésével.

A JavaScriptre biztonsági okokból minden böngészőben vonatkozik a *same domain policy*, amely azt jelenti, hogy csak azon domainhez tartozó felekkel kommunikálhatnak, amelyekről maguk a szkriptek is származnak, ez egy másik technológia segítségével azonban megkerülhető. Amennyiben engedélyezzük, hogy a szerverünkön található információt egy másik helyről letöltött szkript is elérhesse, ún. *cross-origin resource sharing (CORS)* [14][15] megoldást alkalmazunk.



A kliens és a számításokat koordináló szerver kapcsolata

A CORS működéséhez a BOINC szervert is futtató állomás webkiszolgálójában az engedélyezni kívánt tartalmakat el kell látni egy engedélyező *headerrel*, amelyben megadjuk annak a hostnak a címét, amely számára elérhetővé kívánjuk tenni adatainkat. Apache szerver esetén a műveletet *.htaccess* fájlok segítségével végezhetjük el. Az eljárás a BOINC szerver számára teljesen átlátszó, annak fájljait módosítani nem kell.

3.3. Adattárolás

A felhasználói beállítások és a számítások köztes eredményeinek tárolásához a HTML5 *Local Storage* szabványa nyújt lehetőséget. A szabvány egy domainhez kötött perzisztens tárhelyet specifikál, melyben kulcs-érték párok formájában tárolhatunk adatokat. Mérete jelenleg böngészőtől függően 2-10 MB között mozog, amely a tipikusan számításintenzív feladatok adatainak tárolásához elegendő.

A Local Storage-ot megelőzően a kliens csak *cookie*-k formájában tárolhatott adatokat, amelyeket a böngésző minden HTTP lekérdezés során elküldött a szervernek. A Web Storage JavaScriptes utasítások segítségével kezelhető, a böngésző nem küldi el tartalmát a szervernek, így nem terheli vele a kommunikációt.

3.4. Az alkalmazott technológiák támogatottsága

A kifejlesztett függvénykönyvtár működését lehetővé tévő megoldások szabványos eljárások. A Firefox, a Chrome és a Safari már most teljes mértékben támogatja a felhasznált technológiákat, a szabványosságnak köszönhetően pedig csak idő kérdése, hogy a jelenleg lemaradásban lévő Internet Explorer és Opera is felzárkózzon a többiekhez.

4. A projekt jelentősége és a jövőbeli lehetőségek

A böngészők aktuális fejlettsége révén megoldásunk tapasztalataink szerint tökéletesen alkalmas az internetezők széles rétegének közösségi számításokba való bevonásához.

Egy megfelelő modell segítségével az önkéntes gridek rendelkezésére álló számítási teljesítmény nagyságrendileg növekedhetne. A hagyományos, tudományos webalkalmazáson túl a közösségi oldalakba integrált megoldások és egyéb üzleti modellek is elképzelhetők. A hétköznapi felhasználók által nyújtott egyetlen erőforrásnak eddig a reklámok befogadását tekintették. Pedig a technológiai fejlődés folyományaként a hagyományos tartalmak böngészése közben egyre nagyobb a számítógépek kihasználatlan teljesítménye, miközben a tudományos vagy üzleti kutatások terén mindig lehet hasznosítani több erőforrást.

Az állandó fejlesztéseknek köszönhetően mára eljutottunk oda, hogy a böngészők segítségével a processzorok teljesítményét a natív megoldásokhoz hasonló hatékonysággal tudjuk felhasználni. A grafikus kártyák nyújtotta lebegőpontos számítási erőforrásokat ma még nem tudjuk webes alkalmazások segítségével általános célú számításokra felhasználni, a már bejelentett és fejlesztés alatt álló *WebCL* API azonban ezt is lehetővé fogja tenni.

Az új szabványoknak köszönhetően könnyen megtörténhet, hogy a webes alkalmazások nemcsak a natív programokat kiegészítő korlátozott megoldások lesznek,

hanem az önkéntes közösségi számítások világának vezető szereplőjévé válnak. Erre azért is számíthatunk, mert jelenleg valamennyi piaci szereplő a webes alkalmazások terén próbál sikereket elérni, a Google saját operációs rendszerével egyenesen a böngészőt akarja a programok első számú platformjává alakítani.

Annak érdekében, hogy megoldásaink széles felhasználói bázisra tegyenek szert, a jövőben meg kell fontolni a forráskód formális megnyitását, és gondoskodni kell fenntartható üzleti modellek kialakításáról is. A Web Computing projekttel szeretnénk ennek a feltartóztathatatlanul fejlődő számítástudományi területnek minél nagyobb szegmensét lefedni.

Köszönetnyilvánítás

A munka a *Nemzeti Technológiai Program (*TECH_08-A2/2-2008-0097 WEB2GRID) támogatásával valósult meg.

Irodalom

[1] WebComputing kezdőoldal: http://webcomputing.iit.bme.hu/, 2011. április
[2] Web2Grid projekt: http://web2grid.econet.hu/, 2011. április
[3] Distributed.net projekt: http://www.distributed.net/Main_Page/en, 2011. április
[4] NOW projekt: http://now.cs.berkeley.edu/, 2011. április
[5] SETI@HOME projekt: http://setiathome.ssl.berkeley.edu/
[6] BOINC projekt: http://boinc.berkeley.edu/, 2011. április
[7] BOINC scheduling server protocol: http://boinc.berkeley.edu/trac/wiki/RpcProtocol/, 2011. április
[8] BOINC protocol overview: http://www.boinc-wiki.info/Protocol_Overview/, 2011. április
[9] BOINC Web RPC: http://boinc.berkeley.edu/trac/wiki/WebRpc, 2011. április
[10] HTML5 technológiák támogatottsága a böngészőkben: http://www.caniuse.com/#cats=HTML5,JS_API, 2011. április
[11] Haxe nyelv: http://haxe.org/doc, 2011. április
[12] Haxe API: http://haxe.org/api,2011. április
[13] Haxe language reference: http://haxe.org/ref, 2011. április
[14] Cross-Origin Resource Sharing: http://www.w3.org/TR/cors/, 2011. április
[15] Cross-Origin Resource Sharing: http://en.wikipedia.org/wiki/Cross-Origin_Resource_Sharing, 2011. április
[16] Web Workers: http://www.w3.org/TR/workers/, 2011. április
[17] Same origin policy: http://en.wikipedia.org/wiki/Same_origin_policy/, 2011. április
[18] Usage share of web browsers: http://en.wikipedia.org/wiki/Usage_share_of_web_browsers, 2011. április
[19] Ajax: http://en.wikipedia.org/wiki/Ajax_(programming), 2011. április
[20] JavaScript nyelv http://en.wikipedia.org/wiki/JavaScript, 2011. április