ChessBrain

Ha egyetlen számítógép is megver sakkban, mi lesz, ha 646 fog össze ellened?

Idén májusban 646 gép működött együtt egyetlen sakkparti lejátszásában. Ez volt az első alkalom, amikor ilyen eredményt könyvelhettek el, és ezt a Linux, a nyílt programkód és 37 ország több száz programozójának együttes ereje tette lehetővé.

A ChessBrain (➲ http://chessbrain.net) egy osztott számítási program, ami a hálózaton elérhető gépek kihasználatlan számítási kapacitását aknázza ki nagy számításigényű feladatok megoldására. A ChessBrain-rendszer sakkjátszmák lejátszására összpontosít, de a háttérrendszer más játékokhoz vagy nem játék jellegű feladatokhoz is átalakítható.

Képzeljünk el egy teljesen szokványos játszmát, azzal a különbséggel, hogy az ellenfél minden lépése után telefont ragadunk és elkezdjük fölhívni a barátainkat, hogy adjanak tanácsot. Elmagyarázzuk az állást az elsőként hívott barátunknak, és megkérjük, hogy hívjon vissza, ha van valami ötlete. Azután másvalakitől kérdezzük meg, hogy aggódnunk kell-e egy küszöbön álló támadás miatt, és őt is megkérjük, hogy hívjon vissza, ha megvan a válasz. Valahogy így sakkozik a ChessBrain.

A ChessBrain két része a SuperNode nevű, Linux alapú kiszolgálóalkalmazás és a PeerNode nevű ügyfélprogram. A SuperNode egy hálózati játékkiszolgálóhoz csatlakozik, ami lehetővé teszi, hogy a bejelentkezett tagok egymás ellen játsszanak, kihívják a ChessBraint, vagy figyelemmel kísérjék a ChessBrain és pillanatnyi ellenfele játszmáját. A ChessBrain játék közben megvizsgálja az állásokat, lehetséges lépések százait küldi el feldolgozásra a távoli PeerNode-ügyfelekhez, begyűjti a válaszokat, feldolgozza őket, és meglépi a legjobb lépést. A ChessBrain hálózatba kötött gépek folytonosan változó halmazaként létezik. Ez mind filozófiai, mind tudományos szempontból nézve gyönyörű.

A ChessBrainnel 2001 nyarán kezdtem el foglalkozni mint osztott számítási kísérlettel. Az év végére elkészült egy működő prototípus, szükségem volt tehát egy helyre a kiszolgálóprogram számára. Régi barátom, ***Walter Howard***, a HackerWacker honlap (➲ http://hackerwacker.com) gazdája felajánlotta a saját T1 vonalát a kiszolgáló elhelyezésére.

2002. június 9-én a ChessBrain föltűnt a Slashdoton, a kedvező hírverés hatása pedig több száz új PeerNode-ot futtató fehasználó megjelenésében mutatkozott. Egyikük, ***Gavin Roy,*** a bteg hálózat (➲ http://www.bteg.net) tulajdonosa, felajánlotta, hogy ingyen helyet biztosít a SuperNode-kiszolgálónak. Június 27-én Gavinnel ebédeltem, és átnyújtottam ennek a jószerivel idegen fickónak egy SuperNode-kiszolgálót egy Pentium III gépen. A ChessBrain gazdagabb lett egy kiszolgálóval, én pedig egy baráttal, Gavin ugyanis a ChessBrain egyik fontos támogatója lett. Áthelyeztem a SuperNode-kiszolgálót Gavin oldalára, Walt pedig tovább üzemeltette az eredetit tartalék- és kísérleti kiszolgálóként.

Az ezt követő hónapok során hihetetlen mértékű figyelem irányult ránk. Nem úgy tűnt, mintha bárkit is zavarna, hogy a ChessBrain tulajdonképpen nem tud sakkozni. 2002 első nyolc hónapja a SuperNode-kiszolgálón végzett munkával és a PeerNode-ügyfél Microsoft Windows és Apple Mac OS X rendszerre történő átültetésével telt.

Amikor a kiszolgáló és az ügyfelek jól működtek, a hangsúly arra tevődött át, hogy a ChessBrain képes legyen sakkozni. A hollandiai wbec-ridderkek (➲ http://www.wbec-ridderkek.nl) honlap közel 200 szabadon elérhető sakkprogramot sorol föl. Átnéztem néhányat, viszonylag tiszta és több operációs rendszeren fordítható kódot keresve. Tökéletes programnak bizonyult a Beowulf, amit ***Colin Frayn*** írt, aki akkor az angliai Cambridge Egyetem doktorandusza volt. Váltottunk jó néhány levelet, és Colin beszállt a fejlesztésbe. Végig hálózaton keresztül folytattuk az együttműködést, elektronikus levelek és csevegőüzenetek (instant messaging) segítségével, és elkezdtük a szükséges változtatásokat. Colin osztott számításhoz igazította a sakkprogramját, én pedig módosítottam a SuperNode és PeerNode programokat, hogy használhassák a sakkprogramot. A London és Los Angeles közötti időeltolódás éppen megfelelt nekünk. Üzentem Colinnak délután háromkor, és utána a nap folyamán. Az én délutánom közepe felé Colin lefekvéshez készülődött, és én végigdolgoztam az ő éjszakáját. Összeomlás előtt küldtem neki valami visszajelzést. Ez az egész napos ciklus hónapokig folytatódott.

1. kép Az első ChessBrain-kiszolgálók

Colin eredeti Beowulf sakkprogramjából két sakkozó összetevőt hozott létre BeoServer és BeoClient néven. Úgy alakította ki ezt a párost, hogy a ChessBrain keretein belül játszhassanak le sakkjátszmákat. 2002. december 22-én a ChessBrain lejátszotta első osztott számítással végzett sakkpartiját. 2003 januárjára a ChessBrain-közösség 62 gépet üzemeltetett és rendszeresen tesztelte az új programváltozatokat.

Áttekintés

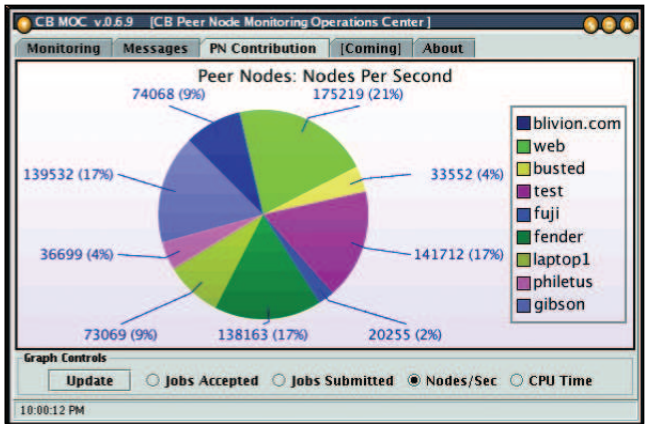
A SuperNode és a PeerNode többszálas C++-program, amit GCC-vel fordítottunk Red Hat Linux 7.1, 7.2 és 8.0 rendszeren. Az elsődleges SuperNode-kiszolgáló Slackware 8.0 alatt fut a bteg hálózat észak-kaliforniai telepén (*1. kép*).
Mivel az alkalmazások erősen többszálasítottak, sok időt kellett a szálakkal kapcsolatos gondok megoldására fordítanom. A GDD és a DDD mellett egyedi naplózással kezeltem a hibakeresési feladatokat. A fejlesztési folyamat korai szakaszában a Perl-parancsfájlok különösen alkalmasnak bizonyultak az új szolgáltatások kipróbálására és a program terheléses tesztelésére. 12 gépem van otthon, ezek egy seregnyi, a helyi kiszolgálót bombázó Perl-parancsfájllal tekintélyt parancsoló teszteszközt alkottak.

XML, SOAP és webszolgáltatások

A fejlesztés kezdetén világos volt, hogy a SuperNode kiszolgálónak kapcsolatot kell tartania más kiszolgálókkal. Akkoriban az XML megfelelő megközelítési módot jelentett, és később az XMLRPC (http://www.xmlrpc.org) további előnyöket nyújtott. Az egyszerű objektumelérési protokoll (Simple Object Access Protocol, SOAP) a más kiszolgálókkal beszélő kiszolgálók igényének kielégítése irányába fejlődött. A kedvezőbb együttműködés lehetősége bátorított arra, hogy a SOAP legyen a SuperNode-kiszolgáló és a PeerNode-ügyfelek közötti kapcsolattartás eszköze. Kívülről nézve a SuperNode úgy működik, mint egy webkiszolgáló SOAP alapú csatlakozófelületekkel. Bár a SuperNode-kiszolgáló GET- és POST HTTP-üzeneteket is használ, a POST a gyakoribb. A SuperNode HTTP és XML alapú SOAP-kéréseket fogad, feldolgozza őket, és HTTP-csomagokat küld vissza, beágyazott SOAP kihelyezésekkel.
A SuperNode és a PeerNode feldolgozza a SOAP-kéréseket, és a parancsokat egy belső parancselosztónak továbbítja, ami biztosítja, hogy a megfelelő parancskezelő dolgozza fel a kérést. A SuperNode-ban a leggyakoribb kérések a PeerNode-ügyfelektől érkeznek; a PeerNode-nak kapcsolódnia kell, hogy lekérhesse a következő feldolgozási egységet. A feldolgozási egység olyan XML-blokk, ami egy játékállást ír le, és utasításokat ad arra nézve, hogy az állást hogyan kell elemezni.
A PeerNode egy teljes sakkprogrammotort tartalmaz, amit statikus könyvtárként fordítunk. Amikor a PeerNode megkapja a feldolgozási egységet, értelmezi a SOAP-választ, kibányássza a feladatra vonatkozó adatokat és utasításokat küld a sakkmodulnak.
Ezután a SuperNode-kiszolgáló a pillanatnyi állást elküldi a külső BeoServer folyamatnak. A folyamatközi kapcsolattartás két cső segítségével valósul meg. A közeljövőben a BeoServer terveink szerint külön gépre kerül, és UDP 1000Base-T ethernet alkalmazására állunk át.

Biztonság

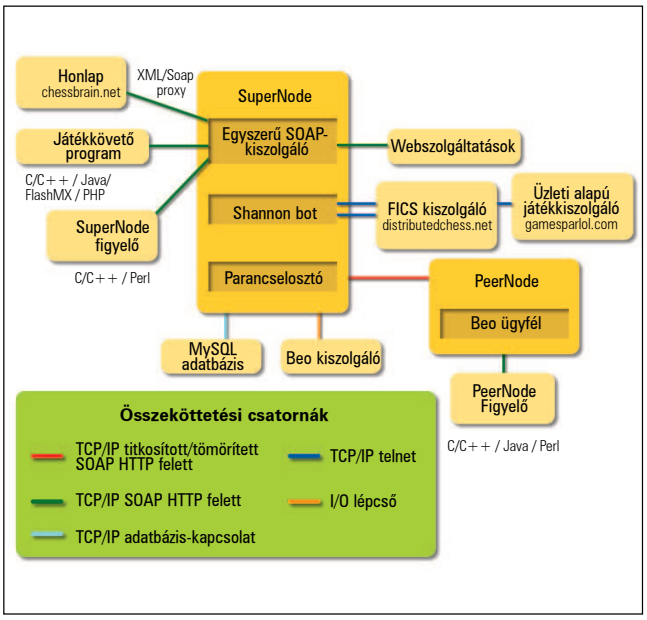
A biztonságos és hiteles üzenetváltás lényeges a ChessBrain számára. Egy rosszindulatú felhasználó egy meghamisított eredménnyel elronthatná a játékot és kínos helyzetet teremthetne. Az érzékeny adatokat a továbbítás során az AES Rijndael nevű fejlett titkosítási szabvány (Advanced Encryption Standard) védi. Az AES a belga ***Joan Daemen*** és ***Vincent Rijmen*** által kifejlesztett változó blokkhosszúságú, szimmetrikus titkosítási algoritmuson alapul, aminek a kiöregedő DES és háromszoros DES szabvány kiváltása a célja.
Mielőtt a Rijndaelt felfedeztük volna, a Blowfish szimmetrikus kódolót használtuk, de a PeerNode Mac OS X rendszerre történő átültetésekor az általunk használt Blowfish-megvalósításban gondok merültek fel a bájtsorrend körül. Az AES algoritmus működése nem függ a bájtsorrendtől, így az adott helyzetben ez volt a legjobb megoldás.
A PeerNode eredeti felépítése szerint két külön folyamatban futtatta az ügyfélprogramot és sakkprogramot. A PeerNode indította a sakkprogramot, és a szabványos kimenet átirányításával hozott létre laza kapcsolatot. Eredetileg el akartuk kerülni a sakk-kód beültetését a PeerNode-ügyfélbe, hogy a program későbbi változataiban könnyen és gyorsan lecserélhessük. Később biztonsági megfontolásból statikus csatolásra tértünk át. A gondot az okozta, hogy lehetett olyan sakkproxy programot írni, ami a PeerNode és a valódi sakkprogram között helyezkedik el. Ez könnyen lehetővé tette volna az eredmények megváltoztatását, mielőtt a SuperNode-kiszolgálóra



2. kép Java alapú CBMoc

3. kép Valós idejű OpenGL-leképezés

kerülnek. A statikus csatolás mellett két fontos előnye miatt döntöttünk: az egyik a nagyobb biztonság, a másik pedig a függvényalapú adatátadás az I/O alapú helyett.
A Slashdotról kiinduló érdeklődéshullám szükségessé tette a ChessBrain sávszélességigényének a korlátozását. Ebből a szempontból nézve, bár a SOAP számos előnyös tulajdonsággal bír, a mérete némi kívánnivalót hagy maga után.
Ma már a ***zlib*** tömörítőkönyvtárat (➲ http://www.zlib.org) használjuk a titkosítás előtt a SOAP alapú üzenettovábbítás méretének csökkentésére. A tömörítés és a titkosítás korlátozza az együttműködési lehetőségeket, ugyanakkor az XML-tömörítési szabvány (➲ http://www.w3.org/TR/xmlenc-core) egy másik megközelítési módot kínál.



A ChessBrain-rendszer felépítése

1. lista A ChessBrain XML-válaszcsomag

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env=
  http://www.w3.org/2001/12/soap-envelope
  xmlns:enc=
    "http://www.w3.org/2001/12/soap-encoding">
  <env:Body>
    <cbs:CBSGetPosResponse

xmlns:cbs="http://www.chessbrain.net">
      <return>
        rn1qk2r/p2p1ppp/bb2pn2/1p6/1P6/
        P2Q1NP1/1BP1PP1P/RN2KB1R b KQkq -
      </return>
    </cbs:CBSGetPosResponse>
  </env:Body>
</env:Envelope>
```

2. lista Egy XML kiszolgáló-állapotjelentés

```
<?xml version="1.0" ?>
<env:Envelope
xmlns:env="http://www.w3.org/2001/12/
                ➥soap-envelope"
xmlns:enc="http://www.w3.org/2001/12/
                ➥soap-encoding">
  <env:Body>
    <cbs:CBSSysInfoResponse
      xmlns:cbs="http://www.chessbrain.net">
    <Uptime days="1" hours="14"
            minutes="43" seconds="18" />
    <System proccnt="546" totmem="250.13"
            freemem="4.38"
            memu="98"
            cpustates="3627078,0,2151891,
            8160852"
            loadavg="0.50,0.30,0.33" />
      <Recv Bytes="2301480887.000000"
            Packets="16652816.000000"
            Errors="0.000000"
            Drop="0.000000" />
      <Send Bytes="2443488824.000000"
            Packets="12142245.000000"
            Errors="0.000000"
            Drop="0.000000" />
    </cbs:CBSSysInfoResponse>
  </env:Body>
</env:Envelope>
```

Botok, jelenlét és önálló játék

A SuperNode-kiszolgálónak van egy Shannon nevű botja (szálként megvalósítva), ami kapcsolódik az internetes játékkiszolgálókhoz és fönntartja a ChessBrain jelenlétét. A játékkiszolgáló felhasználói parancsokat írhatnak be, amikkel a ChessBraint kihívják, vagy figyelemmel kísérhetik a játékot. Jó szórakozás volt a Shannon programozása, és a bot ma már sokféle parancsot ismer. Nagy lehetőségek rejlenek a felhasználói bevitel helyett alkalmazott hálózati botokban. A ChessBrain fejlesztése során letöltöttem a Free Internet Chess Server (FICS) forráskódját, és lefordítottam egy Linuxot futtató régi Pentium 200 MMX Toshiba laptopon. A FICS C nyelvű, és a GCC gond nélkül fordította le. A kiszolgáló lehetővé teszi, hogy a felhasználók `telnet`-tel kapcsolódjanak az 50000-es kapura, és felhasználói név és jelszó segítségével bejelentkezzenek. Néhány hónap múlva a forgalom növekedése miatt a FICS kiszolgálót egy másik, a ➲ http://distributedchess.net tartományon elhelyezett ChessBrain-kiszolgálóra vittük át.

A felhasználóknak már több lehetőségük nyílik a játszmák követésére. Bejelentkezhetnek közvetlenül a `telnet`-tel a játékkiszolgálóra, ahol a ChessBrain játszik, vagy használhatják valamelyik játszmafigyelő programunkat.

Miután a ChessBrain képessé vált arra, hogy játékkiszolgálókon játszmákat folytasson, írtam egy Java-játszmafigyelőt, valamint PHP és Macromedia Flash alapú figyelőket is (➲ http://www.chessbrain.net/viewers.html). ***Anthony Bravo*** írt egy Java nyelvű hálózati figyelőprogramot, ami a világszerte éppen működő PeerNode-okat mutatja. A felhasználók a csomópontokra kattintva megtudhatják, hány aktív ügyfél van az adott országban.

A ChessBrain-honlapon található összes játszmakövető program a SOAP segítségével tart kapcsolatot a SuperNode-dal. Biztonsági megfontolásból az olyan bővítmények, mint a Java és a Macromedia Flash ActionScript nem engedélyezik, hogy a program más kiszolgálóhoz csatlakozzon, mint amelyikről letöltötték. Ezt az akadályt egy egyszerű XML proxy parancsállománnyal kerültem meg, ami HTTP GET-kéréseket fogad az egyik kiszolgálón, és az ügyfél nevében csatlakozik a SuperNode-kiszolgálóra. Ha például le akarjuk kérni a pillanatnyi játékállást a SuperNode-kiszolgálótól, a következőt írjuk a böngésző címsorába:

```
http://www.chessbrain.net/
➥xmlproxy.php?command=CBSGetPos
```

A kiszolgáló válaszul egy SOAP-csomagot küld, amilyet a fenti *ábrán* is láthatunk. A Mozilla böngészőben az oldal forráskódját megnézve látható a SOAP állomány.

A SuperNode megfigyelése

Egy kiszolgáló egészségi állapotának nyomon követése fontos része a rendszerfelügyeletnek. A fejlesztők szerencséjére a Linux sokféle módon megkönnyíti a kiszolgálók megfigyelését. A Linux */proc* virtuális fájlrendszere valóságos aranybánya az értékes rendszeradatok szempontjából, ami a fejlesztők számára könnyebbé teszi a rendszerműködés beállítását és nyomon követését. A */proc/net/dev* olyan adatokat mutat, mint a hálózati illesztőn küldött és fogadott bájtok és csomagok száma, a */proc/meminfo* pedig bőséges statisztikával szolgál a memóriahasználatról. Aki az adatokat nem szereti a */proc* fájlrendszerből kibányászni, a `sysinfo()` segítségével könnyen és gyorsan tölthet fel adatszerkezeteket olyan rendszerstatisztikákkal, mint a rendszerterhelés, a szabad memóriaméret és a futó folyamatok száma.

A SuperNode-kiszolgálótól a 2. *listán* látható példához hasonló rendszeradatok kérhetők le egy SOAP-hívással. A ChessBrain-közösség egyik tagja, ***Greg Davis*** írta az első programot a SuperNode megfigyelésére Perl nyelven, ami a SOAP-kérésre kapott választ a `top` parancshoz hasonló formában jeleníti meg.

A PeerNode megfigyelése

Mivel a ChessBrain-közösség számos tagja PeerNode-ügyfelet futtat sok gépen, egyszerű módon szerették volna nyomon követni egy gépcsoport állapotát. A PeerNode-ügyfelet úgy módosítottuk, hogy a 3434 kapura SOAP-kéréseket küldjön, így az ezen a kapun figyelő programok valós idejű állapotjelentést jeleníthetnek meg. Az első PeerNode-megfigyelőprogramot én írtam, aztán mások is elküldték saját változatukat. Grag Davis és ***Oliver Otte*** egy-egy Perl alapú programot tett közzé.

A legnépszerűbb PeerNode-megfigyelőt, a CBMoc programot ***Kris Drent*** írta Java nyelven.

Hordozható grafika és adatmegjelenítés

A ChessBrain rengeteg adatot gyűjt össze, és jelenleg is dolgozunk azon, hogy az adatok megjelenítési módját hasznosítsuk a rendszerfigyelés és -elemzés során. 3D-s navigációs eszközöket fejlesztünk, amik lehetővé teszik a hálózati játékmenet nyomon követését. ***Sven Herrmann***, házi 3D-szakértőnk OpenGL alapú leképezőt hozott létre, amit a SuperNode-megfigyelő programunkban használunk. Ezt a leképezőt használja majd a képernyővédőnk és a játszmakövető programjaink következő nemzedéke is.

Összegzés

Jelenleg a ChessBrain egy működő prototípus, ami több száz gépen futtatva sakkozik Linux, FreeBSD, Mac OS X és Microsoft Windows operációs rendszeren. Nemzetközimesterszinten játszik, és számos lehetőséget látunk a további fejlesztésére.

A ChessBrain jó példa arra, hogyan oldható meg egy összetett feladat nyílt forrású eszközökkel. Arra készülünk, hogy a ChessBrain kódját is nyílt forrású modellben fejlesszük tovább, azt remélve, hogy mások is csatlakoznak erőfeszítéseinkhez.

A ChessBrainben az a csodálatos, hogy annyira sokféleképpen lehet hozzájárulni. Bárki részt vehet benne, akinek van egy gépe és internet-hozzáférése. Elég letölteni egy PeerNode-ügyfelet a ChessBrain-honlapról, futtatni egy vagy több gépen, és a ChessBrain máris nagyobb lett.

Arra törekszünk, hogy a „legtöbb, egyetlen játékot játszó számítógép” hivatalosan elismert világrekordját állítsunk fel. Már kapcsolatba léptünk a világrekordokat nyilvántartó londoni irodával és több hivatalos sakkszövetséggel is.

A ChessBrain mögött erős csapat áll, többek között ***Peter Wilson***, a sakk világszövetség (FIDE) számítógépes sakk- és internetbizottságának korábbi elnöke. Pillanatnyilag megfelelő helyszínt keresünk egy nyilvános és internet alapú bemutatóhoz, ami magában foglalná az osztott számításhoz és sakkhoz kapcsolódó Guinness világrekord felállítását is. Érdemes figyelni a ChessBrain-honlapon a hivatalos bejelentést.

Köszönetnyilvánítás

Szeretném megköszönni ***Janus Daniels***, ***Cedric Griss*** és a ChessBrain-közösség más tagjainak a támogatását, amit a cikk írásához nyújtottak. Az elérhetőségek és a ChessBrain-kikicsoda megtalálható a
➲ http://www.chessbrain.net/friends.html weboldalon.

Linux Journal 2003. szeptember, 113. szám

Carlos Justiniano (cjus@chessbrain.net)
A programipar húszéves tapasztalattal rendelkező veteránja. A ChessBrainben számos jelmondatát átülteti a gyakorlatba: „Fogadd el a bonyolultságot!”; „Ne kézenfekvő feladatokat keress!” és „Oldd meg nyílt forráskóddal!”.