Építsünk bioinformatikai célú szuperszámítógépet

Az OSCAR fürtöző környezetben futó bioinformatikai eszközök segítségével 17 újrahasznosított személyi számítógépünket felhasználói lekérdezések teljesítményét növelő rendszerré változtattuk.

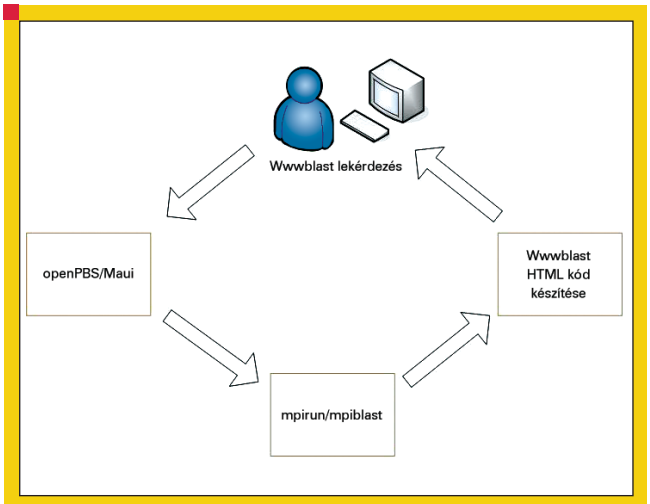
A bioinformatika egyre növekvő jelentőségű tudományterület, melynek egyik ágazata a *DNS* és fehérjeszekvenciák elemezése. A *National Center for Biotechnology Information (NCBI)* által kifejlesztett *Basic Local Alignment Search Tool (BLAST)* az ilyen sorozatok elemzésében segíti a tudósokat. Az eszköz nyílt változata a hálózaton elérhető vagy letölthető. Tekintve, hogy a *BLAST* weblapja igen népszerű, teljesítménye legalábbis hullámzónak nevezhető. A *Dél-Dakotai Egyetem (USD) Bioinformatikai Számítástudományi Csoportja* úgy döntött, hogy nyílt forrású eszközöket felhasználva megpróbálja kialakítani a *BLAST* párhuzamos változatát egy *Linux* fürtön.
A *BLAST* fürt amelybe kivénhedt asztali személyi számítógépek kerültek, felgyorsítja a kereséseket azáltal, hogy a kutatók kis csoportja számára naprakész adatbázist biztosít.
A fürt megoldásunk alapozását az *Open Source Cluster Application Resources (OSCAR)* alkalmazással kezdtük. Az *OSCAR* rendszert az *Open Cluster Group* tervezte fürtözött számításokhoz azzal a céllal, hogy a *Linux* alatti fürtkészítéshez valamennyi eszköz egyetlen csomagban elérhető legyen. Az *OSCAR* segít leegyszerűsíteni a telepítést, a karbantartást, sőt még a fürtprogram használatát is. A grafikus felhasználói felület lépésenként végigvezet bennünket a folyamaton majd később grafikus karbantartó eszközként használható.
Az *NCBI* által készített *WWW BLAST* web alapú kezelőfelület, megkönnyíti a *BLAST* felhasználók dolgát.
Ezt a rendszert választottuk mi is a *BLAST* fürtünkhöz.
A *WWW BLAST* könnyedén telepíthető minden webkiszolgálót (például *Apache*) futtató *Linux* gépre.
Míg a *WWW BLAST* a fürt használati értékét növeli meg, az *mpiBLAST* a teljesítményt fokozza.
Az *mpiBLAST* a *Los Alamos National Laboratories (LANL)* fejlesztése, melynek célja a *BLAST* teljesítményének növelése a lekérdezések párhuzamos végrehajtása által. Az *mpiBLAST* szíve a párhuzamos programokhoz gyakran használt az *üzenetközvetítő felület (Message Passing Interface, MPI)*. Az *mpiBLAST* minden programfeltételt biztosít számunkra a párhuzamos *BLAST* lekérdezések futtatásához.

Lekérdezések áttekintése

Fürtünkön a *BLAST* keresés a webalapú űrlappal indul. Alapértelmezés szerint a *WWW BLAST* nem támogatja a kötegelt végrehajtást és a feladatütemezést. Szerencsére az *OSCAR* csomag *OpenPBS* és a *Maui* eszközei képesek megbirkózni a feladatütemezés és a terhelésmegosztás problémájával. Ilyen támogatással a fürt már sokkal könnyebben ki tud szolgálni nagyobb közönséget is.
Az *OpenPBS* egy rugalmas köteg sorkezelő eszköz amelyet eredetileg a *NASA* számára fejlesztettek ki. A *Maui* az *OpenPBS* képességeit bővíti kiterjedtebb folyamatvezérlést és ütemezési házirendeket nyújtva.
Miután a felhasználó kiadta a lekérdezést, a *WWW BLAST Perl* parancsfájlja indul el. Ez a parancsfájl az űrlap adatai alapján egyedi munkafolyamatot hoz létre. A munkafolyamat egy program vagy feladat amely az *OpenPBS*-hez kerül végrehajtásra. A munkafolyamat átadása után az *OpenPBS* meghatározza a csomópontok elérhetőségét majd az ütemezései rendszabályok alapján végrehajtja a munkafolyamatot. A munkafolyamat elindítja a *helyi elérésű multiszámítógép (Local Area Multicomputer, LAM)* programot, amely lényegében egy felhasználói szintű, démon alapú, futásidejű környezet. A *LAM* az *OSCAR* telepítés része és *MPI* programok legtöbb szolgáltatását tartalmazza.
Az *OpenPBS* a munkafolyamatot az `mpirun` parancs segítségével indítja el, amely minden csomóponton elindítja a keresést, majd összegyűjt az eredményt. A *WWW BLAST* az eredményeket visszaküldi a böngészőnek, majd a felhasználó számára jól olvasható jelentést készít (*1. ábra*).
A párhuzamos *BLAST* kereséseket megvalósító fürtözési megoldás kialakítása némi programmódosítást igényelt. Sok általunk használt eszköz az alapértelmezett beállításokkal működik, ám a párhuzamos *BLAST* fürt már egyedi beállításokat igényel.

Fürtépítés az OSCAR-ral

A fürtöket többféle személyi számítógépből is elkészíthetjük. A mi 17 csomópontunk 533MHz-es *Intel Celeron* processzorral, 256MB RAM-al és 15GB merevlemezzel rendelkezett, ami mai viszonylatban alacsony kategóriásnak számít. A fürtözés kiépítésekor nem követelmény, hogy minden csomópont teljesen azonos alkatrészekből álljon, de mindenképpen



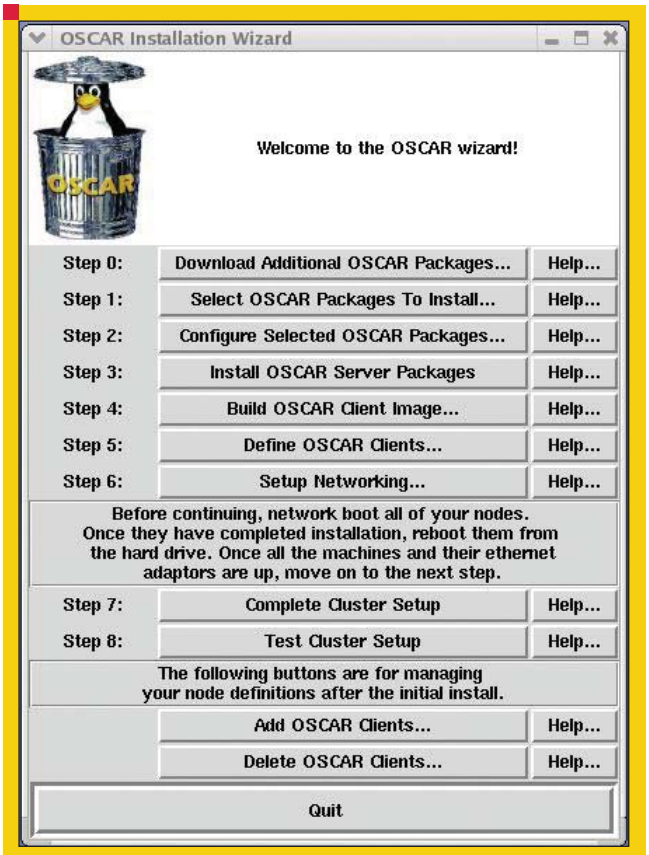
1. ábra Amikor a Webről egy lekérdezés érkezik, a WWW BLAST a munkafolyamatot átadja az OpenPBS-nek. Az OpenPBS az mpirun segítségével elindítja azt, majd a WWW BLAST megformázza a végeredményt.

csökkenti a fürt előkészítésére és karbantartására fordított időt és energiát. Miután a gépeket előkészítettük, ki kell választanunk a fejcsomópontként üzemelő gépet. Amennyiben nem azonos gépeket használunk, nem árt ha a legerősebbet tesszük meg fejcsomópontnak. Tekintve, hogy mi teljesen egyforma gépeket használtunk, véletlenszerűen választottuk ki a fejcsomópont gépet.
Miután beszereztük a szükséges gépi alkatrészeket, ***Linux*** terjesztést kell választanunk. Az ***OSCAR*** dokumentációja felsorolja valamennyi támogatott operációs rendszert, mi a ***Red Hat 9.0***-ás változatot választottuk. A ***Red Hat*** telepítése igen egyszerűnek bizonyult, mindenhol az alapértelmezett beállításokat választottuk. Minthogy az ***OSCAR*** program adott OS csomagoktól függ, az operációs rendszer frissítése nem javasolt az összeállítás után. Természetesen ennek komoly biztonsági vonzatai lehetnek, pontosan emiatt kell a fürtünket az internettől tűzfallal elválasztanunk.
Miután a ***Red Hat*** rendszert feltelepítettük a fejcsomópontra, letöltöttük az ***OSCAR 2.3.1 tar*** csomagját. A telepítési útmutatót a hálózati források között megtaláljuk. Az ***OSCAR*** állományát a root felhasználó saját könyvtárába töltöttük le, hiszen az ***OSCAR***-t root-ként kell majd telepíteni.
Az ***OSCAR*** telepítése mindössze a következő parancsok futtatásából állt:

```
tar -xvfz oscar-2.3.1.tar.gz
cd oscar-2.3.1
./configure
make install
```

A telepítés után a fejgépen valamennyi ***Red Hat 9.0*** RPM-et át kellett másolnunk a ***/tftpboot/rpm*** könyvtárba. Az ***OSCAR*** telepítő bizonyos csomagokat ebből a könyvtárból tölt le a telepítési folyamat során. A következő parancsokkal másoltuk át az állományokat:

```
cp /mnt/cdrom/RedHat/RPMS/*.rpm /tftpboot/rpm
```



2. ábra Az OSCAR segít elhelyezni, beállítani és tesztelni a fürtprogramot

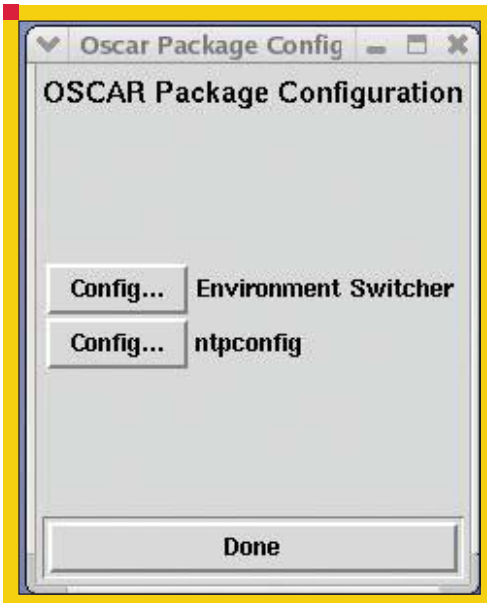
Miután valamennyi RPM-et átmásoltuk, nekifoghatunk az ***OSCAR*** telepítésének. Az ***OSCAR*** grafikus telepítővarázslóval segíti a munkánkat. Helyettesítsük be saját belső hálózatunk ***Ethernet*** csatolóját; a mienk az `eth1` volt:

```
cd $OSCAR_HOME ./install_cluster eth1
```

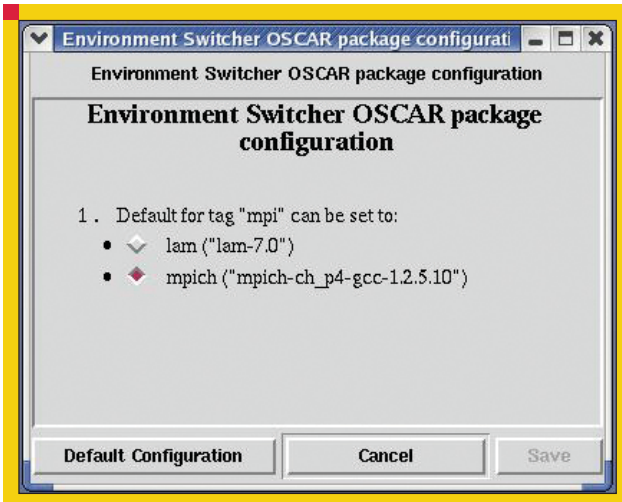
Néhány másodperc múlva az ***OSCAR*** telepítővarázsló megkezdi a betöltést. A varázsló grafikus felületen nyolc lépéses folyamaton végigvezetve segít meghatározni a fürt beállításokat (*2. ábra*). A telepítésünkben az egyetlen eltérés az volt, hogy az alapértelmezett ***MPI*** megvalósítást ***LAM/MPI***-ra állítottuk az ***MPICH*** helyett. Azért választottuk a ***LAM***-ot, mert az ***mpiBLAST*** helyes végrehajtásához erre volt szükségünk. A 2. pontra ***(Configure Selected OSCAR Packages)*** kattintva egy kis űrlaphoz jutunk (*3. ábra*). Itt választhatjuk ki az ***Environment Switcher*** gombot és állíthatjuk be a ***LAM*** rendszert a telepítés alapértelmezettjeként (*4. ábra*).
A további lépéseket az ***OSCAR*** dokumentációnak megfelelően követtük és elkészítettük a csomópontokra szánt telepítőlemezeket. Miután minden csomópontot feltelepítettünk és kipróbáltuk a rendszert, letöltöttük és feltettük az ***mpiBLAST***-ot.

Az mpiBLAST telepítése párhuzamos keresésekhez

Letöltöttük a ***mpiBLAST***-ot, és a ***README*** állományban olvasható dokumentációnak megfelelően telepítettük.
Az ***mpiBLAST*** beállításához mást nem is kellett tennünk, mindössze az `mpiblast` és `mpirun` programokra mutató



3. ábra A beállításokra kattintva... állíthatjuk a környezetet LAM-ra



4. ábra LAM lesz az alapértelmezett környezet

közvetett hivatkozásokat kellett elkészítenünk, úgy, hogy a `$PATH` változó szerint elérhetőek legyenek.
Az *mpiBLAST* telepítése után le kellett töltenünk az adatbázis keresőt. Hogy az *mpiBLAST* helyesen fusson le az adatbázisnak *FASTA* formátumban kel lennie. Az *NCBI* minden adatbázisát összefoglalja *NCBI* weblapon található indexben, ahol a *FASTA* alkönyvtárban valamennyi adatbázist megtalálhatunk *FASTA* formátumban. Letöltöttük az *nr* adatbázis másolatát a */usr/local/mpiBLAST/db/ NFS*-megosztású könyvtárba, amelyet az *mpiBLAST* beállítása során készítettünk el.
Az *mpiBLAST* egyik parancsa az `mpiformatdb` amely az adatbázist szegmensekké alakítja. E szegmensek szám a fürtünk csomópontjainak számától függ.
Az `mpiformatdb` a létrehozott darabokat a megosztott könyvtárba helyezi el. Ezt a könyvtárat az *mpiblast.conf* állományban adhatjuk meg a telepítés során és valamennyi *mpiBLAST* program ezt használja. Nézzünk egy példát az adatbázis formázására:

```
# /usr/local/mpiBLAST/bin/mpiformatdb -N 16 -i nr
```

Itt a `-N` az adatbázis szeletek számát határozza meg amely általában megegyezik a fürtünk csomópontjainak számával az `-i` pedig az adatbázisformátum nevét határozza meg. Példánkban az *nr* adatbázist 16 külön szeletté alakítottuk. Az `mpiformatdb` nem másolja át a szeleteket a csomópontokra, következésképpen az első lekérdezés kiadásakor igen jelentős többletmunkát jelent amíg valamennyi csomópont lemásolja a maga szeletét. Minden csomópont csak egyszer másol le egy szeletet. Ha azonban a szeletet töröljük a csomópontról, a következő lekérdezéskor ismét átmásolódik.
A fürt kezelésének egyszerűsítése érdekében, készíttetünk egy parancsfájlt, amely letölti az adatbázis legfrissebb változatát, megformázza az `mpiformatdb`-vel, majd kiosztja azt a csomópontoknak egy egyszerű *BLAST* lekérdezés kiadásával. A parancsfájlt `cron` segítségével ütemeztük, hogy hetente induljon el. Amikor végül már képesek voltunk párhuzamos lekérdezéseket végrehajtani a *BLAST* segítségével, a *WWW BLAST*-ról feltelepítettük a web alapú előlapot.

A WWWBlastwrap.pl beállítása

Az *mpiBLAST* segítségével ugyan csak parancssori *BLAST* kereséseket adhatunk ki, ám tartalmaz két állományt (*blast.cgi* és *WWWBlastwrap.pl*) amelyek a web alapú előlappal alakítanak ki kapcsolatot. Ezek az állományok úgy vannak beállítva, hogy együttműködhessenek a *WWW BLAST* rendszerével. Következő teendőnk tehát a *WWW BLAST* letöltése volt a */var/www* könyvtárba, és létrehoztuk a */var/www/blast/* mappát. A *WWW BLAST*-ban néhány dolgot át kellett állítani, hogy a *BLAST* kereséseket párhuzamosan hajtsa végre.
A *WWW BLAST* saját könyvtárat biztosít az adatbázisainak. Mivel mi az *mpiBLAST* segítségével alakítottuk ki az adatbázisokat a *WWW BLAST db/* könyvtárát az *mpiBLAST* könyvtárára kellett irányítanunk. Ezért a *blast/* könyvtár db/ mappáját az *mpiBLAST /db* könyvtárára mutató közvetett hivatkozásként hoztuk létre.
A *WWW BLAST blast.cgi* állománya hajtja végre a *BLAST* lekérdezéseket. Az *mpiBLAST*-ban találunk egy másik *blast.cgi* állományt, amely párhuzamos *BLAST* lekérdezéseket hajt végre a *WWWBlastwrap.pl* segítségével. A *WWWBlastwrap.pl Perl* parancsfájl készíti le az *mpiBLAST* által végrehajtandó lekérdezést.
A *WWWBlastwrap.pl* mindezt egy másik *Perl* parancsfájl alakjában teszi, amelyet a webes űrlapról begyűjtött adatokkal tölt fel. Az *OpenPBS* ezt a parancsfájlt kapja meg.
A *WWWBlastwrap.pl* több feladatot is ellát: egyrészt értelmezi az űrlap adatait, létrehozza a parancsfájlt, amelyet a munkafolyamat sorokat és terheléselosztást végző *OpenPBS* kap majd meg valamint a *BLAST* keresési eredményt böngészőbarát formájúra alakítja.
Mindazonáltal néhány módosítást kell végeznünk a *WWWBlastwrap.pl* állományban, hogy helyesen működjön a mi környezetünkben is. Az első változtatás a globális `$scratch_space` és `$MPIBLASTCONF` változók beállítása volt. Ezt a két változót a parancsfájl teljes futása során használja. A `$scratch_space` a lekérdezés alatt

használt ideiglenes állományok elérési útját tartalmazza. A `$MPIBLASTCONF` az ***mpiBLAST*** beállításállomány abszolút elérési útját tartalmazza. Mindkét könyvtárat az ***mpiBLAST*** telepítésekor állítottuk be. A két változót a következőképen állítottuk be:

```
$scratch_space="/usr/local/mpiBLAST/shared/scratch";
$MPIBLASTCONF="/usr/local/mpiBLAST/etc/mpiblast.conf";
```

A következő változtatást az `if` feltételes elágazásokon kell végrehajtanunk. Ezek az utasítások beégetik a `NUMPROC` környezeti változót az ***nt***, ***nr*** és ***pdb*** adatbázisok esetében. Mivel az adatbázisokat az ***mpiBLAST***-al előformázzuk, a lekérdezésenként felhasznált processzorszám állandó. Az alapértelmezett 20-as értéket 16-ra állítottuk, hiszen mi ennyi processzort alkalmazunk:

```
if($data{'DATALIB'} eq "nt"){
    $data{'NUMPROC'} = 16;
}
```

A parancsfájl további részében találjuk a `ValidateFormData` függvényt. Ez az alprogram biztosítja, hogy a felhasználó a megfelelő adatbázis kiszolgáló párosítást válassza, és 500-as kiszolgáló hibát ad vissza ha nem a megfelelő kombinációt választottuk. Ezt az alprogramot módosítottuk az alábbiak szerint, hogy a `tblastx` program hajtsa végre a lekérdezéseket az nr adatbázison:

```
#### ELÖTTE ####
# a nucleotide adatbázisra kell alkalmazni
if($data_ref->{'DATALIB'} ne "nt"){

#### UTÁNA ####
# a nucleotide adatbázisra kell alkalmazni
if($data_ref->{'DATALIB'} ne "nt" ||
   $data_ref->{'DATALIB'} ne "nr"){
```

Később a parancsfájl elkészíti a ***mpiBLAST*** parancssori paramétereit majd eltárolja őket a `$c_line` változóban. Nekünk a -d kapcsolóhoz átadott paramétert kell megváltoztatnunk, amely a keresendő adatbázisról tájékoztatja az ***mpiBLAST***-ot. Alapértelmezés szerint a ***WWWBlastwrap.pl*** összefűzi az adatbázis nevét a processzorok számával és az eredményt átadja a -d kapcsolóval. következésképpen, ha az adatbázisunkat ***nr***-nek nevezik és 16 processzorunk van, akkor ***nr16***-ot adna át. Valószínűleg azért van ez így, hogy az adatbázis több verziójában is kereshessünk és az ***nr16*** a 16-szeletből álló, míg az ***nr8*** a nyolc szeletből álló adatbázist különbözteti meg. Két lehetőségünk van, vagy ilyen alakúra nevezzük az adatbázisunkat, vagy módosítjuk a parancsfájlt. Tekintve, hogy nekünk mindig csak egyetlen adatbázisverziónk lesz, inkább a parancsfájlt módosítottuk és eltávolítottuk a processzorok számát az adatbázis nevéből. A kód változtatásai összefoglalva a következők:

```
#### ELÖTTE ####
# az mpiBlast my-nak átadtott parancsfájl
# létrehozása
```

5. ábra 17 újrahasznosított PC-t tartalmazó fürtünk lerövidíti a felhasználói lekérdezések válaszidejét


```
$c_line = "-d $data_ref->{'DATALIB'}" .
          "$data_ref->{'NUMPROC'} " .
          "-p $data_ref->{'PROGRAM'} " .
#### UTÁNA ####
# az mpiBlast my-nak átadtott parancsfájl
# létrehozása
$c_line = "-d $data_ref->{'DATALIB'} " .
          "-p $data_ref->{'PROGRAM'} " .
```

A tesztlekérdezések futtatása során néhány `lcl|tmpseq_0: Unable to open BLOSUM62` figyelmeztetést találtunk az ***OpenPBS*** hibanaplójában. Ezeket a figyelmeztetéseket úgy tüntethetjük el, ha a ***BLASTMAT*** környezeti változót a ***BLAST*** mátrixok helyére irányítjuk ezért a következő változtatásokat végeztük el:

```
#### ELÖTTE ####
print SCRIPTFILE '#PBS -e '.
"$data_ref->{'ERROR_LOG_FILE'}\n\n";
print SCRIPTFILE 'if(-e $ENV{PBS_NODEFILE} )
➥{'."\n";

#### UTÁNA ####
print SCRIPTFILE '#PBS -e '.
"$data_ref->{'ERROR_LOG_FILE'}\n\n";
print SCRIPTFILE '$ENV{BLASTMAT} = '.
'"/usr/local/ncbi/data";'."\n";
print SCRIPTFILE 'if(-e $ENV{PBS_NODEFILE} )
➥{'."\n";
```

Az utolsó módosítással a parancsfájl vége felé kerültünk szembe a `HtmlResults` alprogramban. A kód, amely a felhasználót az eredményekhez irányítja az alapértelmezett ***URL***-t használja, mi pedig szinte biztosan nem ezt szeretnénk. Az alapértelmezett ***URL***-t átállítva a webkiszolgálónkra az ügyfél webböngészője meg tudja jeleníteni a ***BLAST*** lekérdezés eredményét:

```
#### ELÖTTE ####
print "Location: https://jojo.lanl.gov/blast/".
"BlastResults/$results_file\n\n";
```

```
#### UTÁNA ####
print "Location: http://domain_name/BlastResults".
"/$results_file\n\n";
```

Összefoglalás és eredmények

Helyi fürtünk az *NCBI* weblapján találhatónál kevesebb párhuzamos felhasználóval és jobb összesített átviteli idővel képes keresni a naprakész adatbázisban. Fürtünk és az *NCBI* weblap között egyszerű falióra módszerrel tettünk különbséget. Nyolc egyszerű, fehérje és *DNS* sorozatokat tartalmazó lekérdezést használtunk. Az időzítőt a lekérdezési weblap űrlapjának elküldése után indítottuk és az eredménye megjelenésekor állítottuk le. Az *NCBI* weblapjának eredményei két hét távlatában erősen változtak. Mind a nyolc eredményt átlagoltuk és összehasonlítottuk a fürt idejével. Az eredményeket azért az elküldés pillanatától a megjelenés pillanatáig mértük, mert a felhasználó is ezzel az értékkel találkozik. A fürtnek átlagosan kevesebb időre volt szüksége a lekérdezés végrehajtásához.

Linux Journal 2005. május, 133. szám

Josh Stroschein (jstrosch@usd.edu) Számítástechnikai valamint bűn- és igazságügyi szakértői diplomáját tervezi mostanában. Josh az USD jóvoltából dolgozhat a fürt projekten. Ezen kívül a Vermillioni (SD) Walton Internet Solutions cégnek is dolgozik.

Doug Jennewein (djennewe@usd.edu) informatikus kutatómérnök aki 1998 óta dolgozik az USD-nél. 2004-ben szerezte meg diplomáját az USD-n. Doug fő kutatási területe a nagysebességű számítások.

Joe Reynoldson (jreynold@usd.edu) a kutatás fő irányítója és útmutatója a Számítástechnikai Tanszéken, és 1994 óta dolgozik az USD-nél. Informatikusi diplomáját 1997-ben szerezte az USD-n. Joe Perl, rendszerkezelés és webfejlesztő órákat ad.

KAPCSOLÓDÓ CÍMEK

- oscar.openclustergroup.org
- oscar.openclustergroup.org/tiki-list_file_gallery.php?galleryId=2
- www.linuxjournal.com/article/7462
- www.ncbi.nih.gov/BLAST
- mpiblast.lanl.gov
- ftp.ncbi.nih.gov/blast/executables/release
- ftp.ncbi.nlm.nih.gov/blast/db/FASTA