Aerodinamika és a nyílt forrású alkalmazások

Steve bemutatja, miként lehet nyílt forrású programokat használni repülőgép-kísérleteknél.

Régen, még a Nyílt Forrású Programok mozgalma, a World Wide Web, a Szabad Program Alapítvány és a GNU megszületése előtt az Unsteady Aerodinamics Laboratory of the National Research Council of Canada alkalmazásában álltam. Feladatom a nagysebességű szélcsatornakísérletek adatainak összegyűjtése volt. Abban az időben a laborban egy különleges valós idejű miniszámítógép, egy Hewlett-Packard HP-1000 F sorozatú működött. Valamikor a nyolcvanas évek elején vagy közepén részt vettem a HP International Users Group tanácskozásán, és egy mágnesszalaggal tértem haza, ami a résztvevők által írt programokat és cikkeket tartalmazta. A szalagot a könyvtárrendszerbe fűztem és miután bepillantást nyertem az indexállományba, úgy éreztem magam, mint a kisgyerek, amikor karácsony este kicsomagolja az ajándékait. Először találkoztam forráskódmegosztással és fogalmam sem volt róla, mi mindent tartogathat a jövő egy ilyen nagyszerű és világos elképzelés számára.

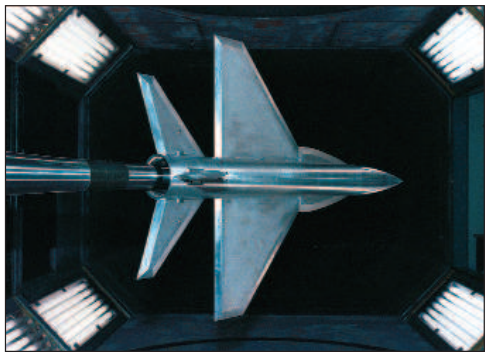
1990-ben kaptam meg az első unixos gépet: az akkor csúcstechnikának számító Silicon Graphics 4D/80GT-t saját T1-es internetkapcsolattal. A kicsi, hálózati kapcsolattal nem rendelkező, valós idejű operációs rendszerrel ellátott gépről az IRIX-alapú hálózati számítógépre történő váltás új, de kockázatos világra nyitott ablakot. Nagyon sok új ismeretet kellett elsajátítanom. 1992-ben vágtam bele a Byzantine nevű parancsállomány megírásába, amely a szélcsatorna adatállományainak kezelésére az `awk` és az `scd` kombinációit használta. Egy napon az egyik feladatom megoldásához tanácsot kértem az Useneten, és valaki megemlítette a Perlt. Bárcsak tudnám, ki volt az – most köszönetet mondhatnék neki, mert sokkal könnyebbé varázsolta az életemet. Egy éven belül a Perl mind az SGI-n, mind az otthoni Macintosh-gépemen nélkülözhetetlenné vált. A Perl a nyílt forrású programok legfontosabb darabja lett a laboratóriumban is.

Abban az időben a Mosaic programmal kísérletezgettem, amit – bár hasznosnak találtam – először igencsak alábecsültem. Nem sokkal később a NCSA által készített HTTPd-t telepítettem, és ekkor csodálkoztam rá a Web lehetőségeire is. A HTTPd később Apache néven született újjá, és a második legnagyobb OSS projektté lépett elő, amelytől egyébként a labor mindennapi munkája is függött.

1995-ben az Aerodinamikai Kutatóintézetet átszervezték, és a zűrzavar csillapultával a mostani helyemen találtam magam: az Aerodinamikai Laboratóriumban, ahol kis sebességű, 23 méteres szélcsatornájának felügyeletét látom el. Akkortájt kezdtem webalapú programokat készíteni, hogy kiterjesszem a már meglévő adatrendszer képességeit. Ezeket parancssoron és QNX, illetve AIX alatt futó X Window rendszeren keresztül lehetett elérni. Azért váltottam böngészőalapú programokra, mert a szélcsatorna adatbázisához kapcsolódó ügyfelek sokszínűsége ezt kívánta meg. Jóllehet munkánk nagy részét repülőgépek kipróbálása tette ki – olyan nagyvállalatok számára, mint a Bombraider Aerospace –, az utóbbi néhány évben azonban autókat, buszokat, teherautókat, motorkerékpárokat (a kedvenceim), villamos távvezetékeket, hidakat, antennákat, valamint olimpiai kerékpárosokat, síelőket és bobcsapatokat is próbára tettünk. Az ügyfelek ilyen széles köre – a számítástechnikához alig értőktől egészen a profikig – igazi kihívást jelentett a felhasználói felület megalkotásában. Mióta a vezetés is tudja, hogyan kell az Interneten keresgélni, elhatároztam, hogy alkalmazásaink közül egyet megpróbálok átültetni Perlre, amelyet azután webalapú felületen lehet elérni. Nagyon kedvező visszajelzéseket kaptam, mind a könnyű használhatóságának, mind az ügyfelek és a munkatársak által tapasztalt magas fokú kénye-

1. kép A kanadai sícsapat tagja 2 m × 3 m-es szélcsatornában

2. kép Jármű vizsgálata 2 m × 3 m-es szélcsatornában

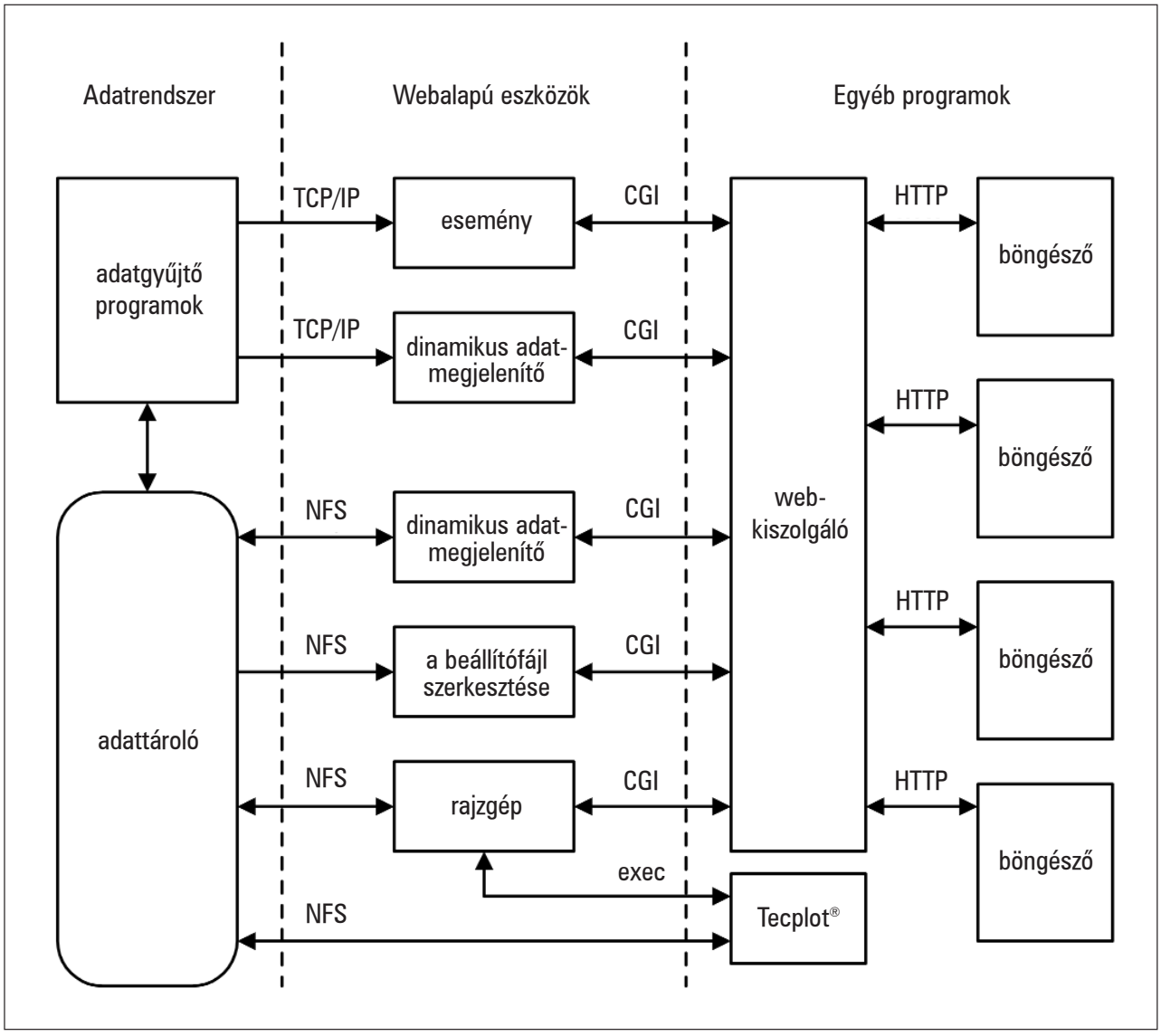


3. kép Harci repülőgép modellje 2 m × 3 m-es szélcsatornában

lemnek köszönhetően, ezért folytattam a webalapú eszközök készítését.
A harmadik kiemelkedő OSS-projektet jó pár évvel később fogadtuk örökbe. 1998-ban egy 24 csomópontos Alpha/Linux Beowulf telepet vásároltunk a folyadékok dinamikájával foglalkozó csoport számítási igényeinek kielégítésére. Ez kiváló lehetőségnek bizonyult az új módszer kiértékelésére, mert a feladat ugyan sokkal számításigényesebb volt, mint a szélcsatorna adatrendszeréé, a labor ügyfelei számára azonban nem bírt akkora horderővel. A sikeres alkalmazás meggyőzött bennünket, hogy az eddig használt kereskedelmi operációs rendszerek mellett a Linux életképes választási lehetőség.
Míg ezek a terjedelmes programok a helyükre kerültek, számos kisebb OSS-programot kezdtünk használni, például a Ghostscriptet, az Xmgr-t, a Vimet és a Neditet.

A jelen

A cikk további követhetőségéhez leírom az egyik, az adatfeldolgozás szempontjából jellemző repülőgép-kísérletet. Miután a modellt elhelyezik a szélcsatornában, 1–5 hét időtartam szükséges a próba befejezéséig. Ez idő alatt több mint ötszázezer mérést végeznek, ennek következtében 2000 X-Y plot, 4000 lemezállomány jön létre, és 500 MB szöveges adat jelenik meg a képernyőkön. Szükségszerű, hogy ennek az adattengernek a kezelésére gyors és egyszerű módszer kell, hogy rendelkezésre álljon. Az ügyfelek és a labor mérnökei az adatok gyűjtését, tárolását és megjelenítőrendszerét minden esetben Apache alatt futtatott Perl CGI programon keresztül érhetik el (lásd *ábránkon*). A szélcsatorna operátorai a kísérlet számos vezérlőelemét ugyanilyen módon kezelhetik. Amennyiben a vezérlőteremben a felhasználó bármelyik számítógépen megnyit egy webböngészőt, a szélcsatorna ügyfélhonlapja önműködően betöltődik. Ezen a lapon keresztül lehet elérni azt az öt webalapú programot, amelyeket az eddigiek folyamán írtam: ábrázoló, beállítási állományszerkesztőt, adatállomány-nézegetőt, eseménynaplózót és dinamikus adatmegjelenítőt. Ezenkívül léteznek hivatkozások a helyi segédeszközökhöz is, például a rendszerleíráshoz, illetve a mértékegység-átváltó számológéphez. Szeretném felhívni a figyelmet, hogy a laboratórium eléggé



Webalapú programrenszer szerkezete

korlátozó típusú helyi hálózati modellt használ, amely segíthet a webalapú rendszereknél előforduló biztonsággal kapcsolatos aggodalmak eloszlatásában. Elsőként az ábrázolórendszert fejlesztettük ki, és annak bizonyítására használtuk fel, hogy a szélcsatorna ügyfelei adataikat a Weben keresztül is el tudják érni. Amikor elkezdtük az Amtek Engineering által készített Tecplot kereskedelmi adatmegjelenítő programot használni, elhatároztam, hogy az ábrázolórendszert e program köré írom meg. A felhasználók az ábrázolómintákat – a beállítások kiválasztásával és szövegmezők kitöltésével – egyszerűen HTML-űrlapon keresztül állíthatják be. Ezeket a mintákat azután Tecplot-parancsállományok előállítására használjuk, ezáltal képernyőn vagy papíron lehetőség nyílik az eredmény megtekintésére. Egy démon (szintén Perlben íródott) ugyanezeket a mintákat használja a nyomtatásra, amely minden szélcsatorna-kísérlet végén önműködően zajlik. A beállítóállomány szerkesztését egy másik webalapú programmal valósítottam meg, amelyben a kísérleti adatokat gyűjtő és egyszerűsítő programok vezérlőállományait gyors és egyszerű módszerrel lehet módosítani. A felhasználók olyan űrlapot látnak, amelynek minden sora szövegmezőket és választógombokat, valamint a hozzá tartozó értékneveket tartalmazza. Az a Perl program, amely ezt a HTML-űrlapot hozza létre, dinamikusan előállít egy JavaScript-kódot is, amelynek az a feladata, hogy az űrlap benyújtása előtt ellenőrizze a kitöltött adatok érvényességét. Ha érvénytelen bejegyzést talál, a beviteli mező mellett villogó nyíl jelenik meg, és egy előugró párbeszédablakban a hiba jellege lesz olvasható.
Az adatállomány-megjelenítő egy egyszerű CGI program, amely egy adott szélcsatornapróba adataiért kutatja át a lemezterületet. Minden bejegyzéshez, amely a keresési mintára illeszkedik, egy HTML-gombot készít. Ezek a gombok táblázatban helyezkednek el, ahol minden sor a hasonló csatornapróbákat, illetve minden oszlop a megadott adattípust tartalmazza (például nyers, egyszerűsített). Bármely gomb megnyomására új böngészőablak nyílik meg, ahol a kiválasztott állomány formázott és elemzett alakban tekinthető meg. Ezután a felhasználók számára adott a lehetőség, hogy a saját gépükre CSV, Matlab vagy bármely más formátumban letöltsék. Minden új alkalmazás és számos régebbi kód állapotüzeneteket állít elő – eseményeket, amelyeket eseménynaplózó rendszerrel kezelünk. Ez a rendszer két fő részből tevődik össze: az első egyszerű Perl-démon, amely egy TCP/IP-kaput figyel, hogy érkezik-e rajta üzenet, s amennyiben igen, a naplóállományokban tárolja őket. A rendszer másik része egy webalapú megjelenítő, amellyel a felhasználók különböző szempontok

4. kép Forgó F18-as modell a vízcsatornában

alapján kereshetnek a naplóállományok között, mint például az esemény ideje, a számítógép neve, az esemény súlyossági szintje. Ez a program jelentéktelennek tűnik, pedig nélkülözhetetlen, mert az adatgyűjtő, -kezelő és -megjelenítő rendszer számos, különböző operációs rendszerrel működő számítógépből áll. Az ilyenfajta osztott rendszerekben a hibakeresés nagyon nehéz (különösen az összetettség miatti összehangolás okoz gondokat), általános eseménynaplózó rendszer nélkül csaknem lehetetlen. A felhasználók szempontjából az egyetlen nem interaktív eszköz a dinamikus adatmegjelenítő rendszer: Perl-kiszolgálón alapul, amely az adatgyűjtőrendszertől adatcsomagokat fogad. A felhasználók egy CGI-programon keresztül a kiszolgálóhoz kapcsolódva tudják megjeleníteni ezeket az adatokat. A CGI-program NPH-t (nonparsed header) vagy „server push”-t használ. A program egy adattáblázatot jelenít meg, amelyben az újabb adatokat dinamikusan a táblázat tetejére írja, a régi adatot pedig lefelé tolja. A program készítése folyamán aggódtam a memóriahézagok miatt, amelyek nemcsak Perlben vagy Apache-ban, de az ügyfelek böngészőjében is előfordulhatnak. Fölöslegesen, hiszen akadtak különleges NPH-ügyfelek, amelyek a kiszolgálóhoz folyamatosan kapcsolódva olyan anyagokért kutatták át a rendszert, amelyek több mint öt hete készültek. Eközben minden gond nélkül 500 MB-nál több adatot jelenített meg.
Ennek az öt programnak bármelyike külön-külön jól használható lenne, de aligha nevezhetők forradalminak. Mihelyt azonban egyesítjük őket, egyszerű, szilárd és nagyméretű környezetet kapunk. Nincs szükség nehezen érthető parancsokra, hosszú adatelérési útvonalakra, parancsbillentyű-kombinációkra vagy bármi másra, amely különféle típusú kezelőfelületekre jellemző. Nem kell mást tenni, mint kattintani, kijelölni, és kitölteni a mezőket – mindenki tudja, mit és hogyan tegyen.

A jövő

Teendőim hosszú listáján előkelő helyen szerepel azoknak a Perl-kódoknak a kétprocesszoros Intel/Linux-rendszerre történő ültetése, amelyeket az utolsó megmaradt SGI-rendszeren fejlesztettem. Bár a programok futtatása a jelenlegi formájukban jelentéktelen feladat, éltem az összes kód újrahangolásának lehetőségével. Még egy alkalmazás befejezése lenne különösen fontos: a modell viselkedését vezérlő rendszer felhasználói felületéé. Ráadásul figyelembe kell vennem adatformátumaink váltását – az arcane házilag fejlesztett formátumától az XML-ig. Ez szintén néhány új kód szükségességét eredményezi. Távolabbra pillantva a jövőbe, remélem, elég időm lesz kifejleszteni néhány VRML-alkalmazást is, amelyek a szélcsatornában elhelyezett modellek és szondák terhelés- és nyomásviszonyait képesek utánozni, és 3D-ben dinamikusan megjeleníteni. A műszeres csoport is vizsgálja a beágyazott, illetve valós idejű Linux-rendszerek felhasználásának lehetőségét.
Mire mindezen munkák elkészülnek, a nyílt forrású programok egyre növekvő fontosságú szerepe már vitathatatlan lesz az Aerodinamikai Laboratórium mindennapi munkájában.

Steve Jenkins az Aerodynamics Laboratory of the Institute for Aerospace Research vezető programozója és elemzője, a légi kutatások adatfeldolgozásában több mint húszéves tapasztalattal rendelkezik.