openMosix: Live free() or die()

Bodnár Csaba

2003.11.08.

Kivonat

Az openMosix napjaink egyik legizgalmasabb Linux-alapú telep-projektje (cluster projekt). Más telep-típusoktól eltérően megpróbálja teljes egészében egyetlen nagy Linux-gépként láttatni a tetszőleges számú csomópontból álló telepet. Az előadás áttekinti az openMosix [1] projekt –az eredeti Mosix szabad leágazása (fork)– jelenlegi állását.

Tartalomjegyzék

1. „Live free() or die()”

1-2 évvel ezelőtt a BSD-ről tartott valaki előadást hasonló címmel. Ennek ellenére én is ezt a címet adtam az előadásnak, egyrészt mert maga az openMosix Howto is ezt a jelmondatot használja, másrészt mert a Mosix „szabadsága” 2001-ben valóban veszélybe került, amikor a projekt két vezetője nem tudott megegyezni annak további licencpolitikáját illetően. Az eredmény a fejlesztés kettéválása lett: Dr. Moshe Bar – mindnyájunk örömére– úgy döntött, hogy továbbra is GPL licenceléssel viszi tovább a kódot, openMosix [1] néven.

2. Az openMosix projekt történetének mérföldkövei

- A 80-as években született PDP-11/70-en. Egy teljes és egy „lemeztelenített” (diszkless) PDP, innen jött a processz migrálás ötlete.
- 1997 óta fut GNU Linux alatt.
- 1999: Az első nagy „pofon” a nyílt forráskódú közösségnek: kijönnek egy csak bináris verzióval...
- 1998 és 2001 között Prof. Barak és Dr. Moshe Bar közösen vezetik a projektet a Hebrew University-n.
- 2001-ben Mosix/openMosix szétválás licenc problémák miatt.
- 2002 júliusára a Mosix installációk 97%-a openMosixra váltott.
- 2002 augusztusa óta a legaktívabb Linux-telep projekt.
- 2002 augusztus: HP-partneri kapcsolat.

3. A különböző telep-típusok áttekintése

Az openMosix egy Linuxos telep. Maga a telep szó (angolul cluster, fürt) nem takar mást, mint hogy valahány számítógépet valamilyen cél érdekében „egybefogunk”, összetartozónak tekintünk. Ha ez a cél a:

- *magas rendelkezésre állás*, akkor magas rendelkezésre állást nyújtó telepről (High Availability, HA, pl. Kimberlite, Red Hat Cluster Manager, ...)
- *terhelésmegosztás*, akkor terhelésmegosztó telepről (pl. LVS)
- *szuperszámítógép teljesítmény elérése*, akkor tudományos-technikai célú telepről (High Performance Technical Computing, HPTC, pl. Beowulf)
- *egy nagy SMP-gépnek látsszon*, akkor SSI-telepről (Single System Image, még nem találtam rá jó magyar kifejezést :) (pl. openMosix, Qlusters [2], OpenSSI)

beszélünk.

4. Mi az openMosix?

Fenti terminológia szerint az openMosix egy olyan SSI-telep, amely elsődlegesen tudományos-technikai feladatok megoldására szolgál. Automatikusan kiegyenlíti a terhelést a telep különböző tagjai között, valamint az egyes tagok menet közben csatlakozhatnak hozzá vagy hagyhatják el a telepet. A terhelést adott szempontok szerint (CPU-sebesség, hálózati sávszélesség, ...) osztja szét a tagok között.

Hasonlóan a klasszikus VMS-telepekhez (a DEC 20-30 évvel ezelőtt csinált elsőként ilyen telepeket VMS operációs rendszer alatt), a telep bármely tagjára belépve a felhasználó által indított processz azon a tagon fog futni, amelyik az adott pillanatban legjobban ki tudja szolgálni annak igényeit, vagyis a rendszer megpróbálja optimálisan szétosztani a tagok között a felhasználók futó processzeit.

Fontos hangsúlyozni, hogy az erőforrások megosztása a processzek szintjén történik, vagyis egy processz egyszerre csak egy teleptag erőforrásait használhatja. A felhasználó úgy jár jól, ha munkáját egymással párhuzamosan futtatható részekre bontja és azokat külön processzként futtatja. A későbbiekben lesz majd lehetőség szálankénti (thread) szétosztásra is.

5. Hogyan működik az openMosix?

Moshe Bar szavaival élve: „A processz migráció maga elég egyszerű, egyszerűen átrakjuk a processzt az egyik gépről a másikra, és –szükség esetén– még több processzt rakunk át. A rendszerhívásokat pedig vagy visszaküldjük az UHN-re (Unique Home-Node), vagy –néhány esetben– helyileg hajtódnak végre (pl. DFSA, közvetlen fájlrendszer elérés esetén). Ez ennél nem lesz sokkal egyszerűbb. Az ördög a részletekben lakozik."

5.1. Az erőforrás-megosztó algoritmusok

Az egyik –az erőforrások felhasználására hatással levő– algoritmus nagyon egyszerű: ha egy adott tagon elkezd elfogyni a memória, a rendszer elkezdi átpakolni a processzeket a többi tagra.

A másik algoritmus dönti el, hogy egy adott processz melyik tagon fusson. Az algoritmus matematikai modellje közgazdasági kutatások eredménye. Annak eldöntése, hogy egy feladatnak melyik tag a legmegfelelőbb helye, igen bonyolult probléma. A legnagyobb baj az, hogy az egy telepen elérhető erőforrások nagyon különbözőek lehetnek: A memória, a CPU, a processz-kommunikáció, stb. nehezen összevethetőek, még csak nem is ugyanazokban az egységekben mérhetőek. Az algoritmus megpróbálja ezeket a heterogén erőforrásokat egységes formára hozni, költséget rendelni hozzá. A feladatok pedig mindig azokon a gépeken fognak futni, ahol a legalacsonyabb a „költségük".

5.2. Processz-migrálás

Minden egyes processznek van egy úgynevezett UHN-je (Unique Home-Node): ez az a teleptag, amelyen a processzt eredetileg elindították. Ezen az UHN-en fut a processznek egy „helyettese" (deputy), amely kapcsolatban van a processz tényleges, valószínűleg valamelyik másik tagon futó példányával, a „távolival" (remote). A legnagyobb probléma a rendszerhívások kérdése: minél több rendszerhívást kell a „távoli-

nak” visszaküldenie a „helyettes” felé, annál lassabb lesz az egész, hiszen a hálózaton keresztüli kommunikáció nagyságrendekkel lassabb a gépen belülinél.

A fejlesztők mindent elkövetnek azért, hogy minél több rendszerhívás tudjon helyileg végrehajtódni: ezt célozta meg az oMFS hálózati fájlrendszer kifejlesztése és a socket migrálás (ha egyszer lesz majd) is.

5.3. openMosix fájlrendszer (MFS), közvetlen fájlrendszer elérés (DFSA))

Az openMosix telep vagy hálózati fájlrendszer kifejezetten a DFSA részére lett kifejlesztve. Segítségével bármely tag el tudja érni a többi tagon lévő fájlrendszerek fájljait, így a migrált processzek anélkül érhetik el a hozzájuk tartozó adatokat, hogy ehhez a rendszernek távoli rendszerhívásokat kellene végrehajtania (nyilván így is lesz hálózati forgalom, de jóval kisebb, mert ez kifejezetten erre a helyzetre van optimalizálva).

5.4. Socket migrálás

Ha a processzel együtt át lehetne küldeni magát a socketet is (vagyis a meglévő TCP kapcsolatot is) valamely másik tagra, az szintén nagyban gyorsítaná a telep működését. Sajnos a megfelelő kódot még nem írta meg senki. . . illetve dehogynem! Amit Shah, az egyik fő fejlesztő saját bevallása szerint tavaly ezen dolgozott, a keletkezett kód azonban a Qlusters Inc. tulajdonába került :(A Qlusters nevű cég egy –az openMosixhoz hasonló– üzleti célú telepszoftver fejlesztésével foglalkozik, techológiai vezetője (CTO, Chief Technology Officer) Moshe Bar, az openMosix projektvezetője. . .

5.5. Telep-szinten elosztott memória (DSM, Distributed Shared Memory)

A fenti Amit Shah instrukciói alapján öt indiai diáklány –a MAASK-csoport [3] – diplomamunkának a DSM létrehozását választotta. Jelenleg ugyanis az openMosix egyik komoly hiányossága, hogy az osztott memóriát használó illetve a többszálú (multithreaded) alkalmazások nem migrálhatók, így nem részesülhetnek annak „áldásaiból”. A MAASK-csoport által fejlesztett Migshm (Migration of shared memory) segítségével mind a shared memóriát használó processzek (shmget(), shmat(), shmdt() and shmctl() rendszerhívások), mind pedig a többszálú processzek (clone() rendszerhívás) migrálhatóvá válnak.

Moshe Bar egyelőre nem tette be a Migshm-et az openMosixba, mondván hogy még nem elég stabil („I would rather have no DSM than bad DSM”).

6. Kezelőfelületek

6.1. karakter-orientált felület

A klasszikus karakter orientált segédprogramok (*mosmon, mosctl, mosrun, moslimit, migrate*) mind rendelkezésünkre állnak, emellett többféle grafikus eszköz is segíti munkánkat.

6.2. OpenMosixView-készlet

Az alábbi X-window alapú segédprogramok OpenMosixView-készlethez tartoznak:

- *openMosixview* - A fő figyelő és adminisztrációs alkalmazás
- *openMosixprocs* - Processz kezelő alkalmazás
- *openMosixcollector* - Telep és teleptag információ gyűjtő alkalmazás
- *openMosixanalyzer* - Alkalmazása a gyűjtött adatok elemzésére
- *openMosixhistory* - Telep-szintű processz-történet alkalmazás

6.3. openMosixWebView

Az openMosixWebView egy web alapú monitorozó eszköz az openMosixhoz.

7. openMosix előnyök és hátrányok

7.1. Előnyök

- Általában nincs szükség az alkalmazás módosítására.
- Nincs szükség kiegészítő csomagokra.
- Egyszerű installálás és konfigurálás (pl. rpm -Uvh openMosix*.rpm).
- openAFS integráció.
- Portolva IA-64-re, AMD-64 port folyamatban.
- Telep-szintű hálózati fájlrendszer (oMFS).
- Több más termék is az openMosixon alapul: openMosixView, openMosixWebView, openMosixApplet, RxLinux, PlumpOS, K12LTSP, LTSP, ...
- Maguk a felhasználók fejlesztik, a felmerülő igényeknek megfelelően.
- Teleptag automatikus felfedezése (Node autodiscovery/fail-over daemon).
- Telep „maszkolás” (Cluster Mask).
- Az automatikus hangolást lehetővé tevő segédeszköz.

7.2. Hátrányok

- Kernelfüggő.
- Telep-szintű osztott memória támogatás még nem stabil.
- Többszálú alkalmazások támogatása még nem stabil.
- Nincs még socket migráció.
- Ha az alkalmazásunk egyetlen processzből áll, nem nyerünk semmit vele.

8. Összefoglalás

Az openMosix már most is egy teljes értékű, sokak által használt SSI-telep szoftver. A várható további fejlesztésekkel kiegészítve nem csak tudományos-technikai számítások, hanem egészen más típusú feladatok (hálózati alkalmazások, adatbázis-kezelők, stb.) terhelésének megosztására is alkalmassá válik.

Hivatkozások

[1] openMosix: `http://openmosix.sourceforge.net/`

[2] Qlusters: `http://www.qlusters.com/`

[3] A MAASK-csoport: `http://mcaserta.com/maask/`