Erdei Csaba
Az openMosix cluster szoftver

1. Számítógéptelepek típusai

Informatikai eszközökkel gyakran speciális feladatokat kell megoldanunk. Ilyen problémák megoldására nyújt jó lehetőséget a géptelepek (cluster, fürt, klaszter) építése, mely során egyedi gépeinket összekötjük valamilyen rendszer szerint, és együtt végzik el a szükséges munkát. E módszer fő előnye a szuperszámítógépekkel, speciális szerverekkel, célgépekkel szemben a lényegesen kisebb költségtényező, mivel az egyre olcsóbb hardverekből (akár PC-kből is) megépíthetőek.

A fürtök által ellátott feladatok alapján az alábbi csoportokat különböztetjük meg:

- High Availability (HA): a rendszer magas rendelkezésre állást biztosít, tipikus felhasználási terület lehet adatbázisszerverek üzemeltetése
- Load Balancing (LB): az eszköz nagy igénybevétel esetén terheléselosztást végez a tagok közt, gyakran webszervereknél találkozhatunk vele
- High Performance Technical Computing (HPCT): nagy számításigényű feladatok elvégzésére alkalmas fürt, ahogy a nevében is szerepel tudományos, technikai célokra
- Single System Image (SSI): egy rendszernek látszó géptelep, amely egyedi gépekből áll, de egyetlen sokprocesszoros gépként viselkedik

Jelenleg hazánkban tudományos és technikai célú feladatok megoldására döntő többségében szuperszámítógépeket és/vagy az említett HPCT típusú géptelepeket használják. Azonban az SSI szoftvereknek vannak olyan tulajdonságaik, amelyek szintén lehetővé teszik az ilyen irányú alkalmazást, illetve az ott található néhány megoldás gondolatébresztőként hathat más területeken is. Egy ilyen, érdeklődésre számot tartó szoftver az openMosix.

2. Az openMosix

Az openMosix tulajdonképpen egy Linux rendszermag-kiterjesztés és néhány felhasználói program, amely lehetővé teszi, hogy gépeinkből SSI cluster-t építhessünk. A program operációs rendszer szinten közösen kezeli az erőforrásokat, és automatikus terheléselosztást végez. Telepítése igen egyszerű. Mivel GNU/GPL licenc alatt fejlesztik, így forrása letölthető és az installálás abból elvégezhető, de néhány rendszeren előre csomagolt formátumban is elérhető (például RPM csomagban).

2.1. Az openMosix története

Az openMosix története az 1980-as években kezdődött PDP-11/70-es gépeken. Egy diskless (merevlemez nélküli) rendszer adta az ötletet arra, hogy egy teljes rendszerről processzek „átrakásával" kihasználják az ott meglévő erőforrásokat is. 1997-ben áttértek a Linux használatára. Ez jó döntésnek bizonyult, hiszen forráskódja nyílt, így alacsony szinten illeszkedő programot lehet fejleszteni hozzá, valamint maga az operációs rendszer gyorsan fejlődik, így az oM fejlesztői csak a saját munkájukkal foglalkozhatnak. 1998 és 2001 között Prof. Barak és Dr. Moshe Bar állt a fejlesztés élén, melynek központja a Hebrew University-n volt. 1999-ben a korábbi verziókkal ellentétben csak bináris formában jelent meg az aktuális verzió, amit nagy szomorúsággal vett tudomásul a szabad szoftveres közösség. Mivel a későbbiekben a projekt vezetői nem tudtak megállapodni a program licencében, így 2001-ben „szétszakadt" a fejlesztés. A Mosix kereskedelmi szoftver lett, míg Moshe Bar vezetésével (aki a mai napig is fő fejlesztő) az openMosix a GPL-t választotta. 2002 óta az egyik legaktívabb linuxos telepprojekt, folyamatosan fejlődik a rendszer tudása és számos segédprogram kapcsolódik hozzá.

2.2. Az openMosix működése

A rendszer működését alapvetően két algoritmus határozza meg. Az egyik az egyes tagokon a memória elfogyását akadályozza meg, míg a másik elosztja az erőforrásokat. E második algoritmus alapelvét közgazdaságtani kutatások szolgáltattak. A rendszer minden tagon feltérképezi a rendelkezésre álló eszközöket és költséget rendel hozzájuk. A fő szempontok természetesen a processzorszám és a sebesség, valamint a memória paraméterei. Ezután a futó processzeket olyan gépekre migrálja, ahol a „legolcsóbb" a futásuk, így törekedve arra, hogy a feladat a lehető leggyorsabban végrehajtódjon. Ez a cluster nem hierarchikus szerkezetű (szemben a HPCT master-node felépítésével), az adott processz áthelyezéséről mindig két tag dönt, majd ha „megéri", akkor a terheltebbről átkerül a program a másikra. A gépek eközben folyamatosan információval látják el egymást terheltségükről, kapacitásukról, státuszukról, ezeket a */proc/hpc* könyvtár alatti fájlokban mi magunk is megnézhetjük.

Processzmigrálásnál a „gazda gépen" (UHN – Unique Home Node) a processzről csak alapinformációk, egy amolyan „szellemfolyamat" marad meg (deputy), maga a kód és az adatok (remote) átkerülnek a másik csomópontra. A deputy tartalmát mi határozhatjuk meg a rendszer beállításánál és ez a folyamat megmarad az UHN processztáblájában. A migrálásnál központi kérdés a rendszerhívások feldolgozása, emiatt van szükség a deputy-ra. A rendszerhívásokat alapesetben a UHN dolgozza fel, ez azonban időkiesést jelent. Emiatt a rendszerhívásoknak minél nagyobb részét megpróbálják elvégezni a távoli gépen, és csak kisebb részét visszaküldeni. Abban az esetben, ha túl sok rendszerhívás van,

akkora lehet a késleltetés, hogy nem éri meg migrálni az adott programot. Ebben az esetben ez természetesen nem is fog megtörténni.

A fájlműveletek gyorsabb feldolgozása érdekében fejlesztették ki az oMFS-t (openMosix File System). Ez egy hálózati fájlrendszer, amelynek segítségével bármelyik tag közvetlenül éri el a többieken lévő fájlokat. Ennek használatával akár az is előfordulhat, hogy nem az adat megy a futó programhoz a hálózaton keresztül, hanem fordítva.

Igen fontos csomag a MigShm rendszerkiegészítő, amely telep szintű osztott memóriát valósít meg (DSM – Distributed Shared Memory), lehetővé téve ezáltal a shared memory-t és socketeket használó programok migrálását. Így akár hálózati alkalmazások és több szálú programok is „áttehetőek" más gépekre. Ez a kiegészítő 2004 közepén került a fő csomagba, előtte az ilyen típusú szoftverek futtatását openMosix használatával nem lehetett gyorsítani. Az SSI fürtünk ilyen tudással már nem csak tudományos technikai célokra alkalmazható, hanem a fent említett HA és LB típusú feladatok végrehajtására is.

Bizonyos esetekben, amikor a telep gépei csak időszakosan használhatóak számítási feladatokra, szükség van a folyamatok leállítására, majd későbbi újraindítására anélkül, hogy az addig elvégzett munka eredményeit elveszítenénk. Erre a CHPOX nevű kernelkiterjesztés nyújt megoldást, amit a kijevi Taras Sevcsenko Egyetemen fejlesztenek. A rendszermagmodul betöltése után signal-ok küldésével menthetjük el alkalmazásaink állapotát bináris fájlba, amely a későbbiekben újraindítható.

2.3. Felhasználói szoftverek

A rendszerhez tartozik ezen kívül még számos felhasználói program, amelyekkel a futó programjainkról, a géptelep állapotáról, illetve csomópontjairól kaphatunk információkat. Karakteres felületű terhelésmegjelenítő szoftver a `mosmon`. Az `mtop` az ismert linuxos `top` parancs openMosix-os megfelelője. Egyszerű konfiguráló alkalmazás a `setpe` és a `mosctl`. Több grafikus felületű segédszoftvert is találunk a projekt weboldalán. A `mosixcollector` a felhasználók által futtatott programok naplózására szolgál, valamint adatot gyűjt a teljes rendszerről. Ehhez hasonló funkciójú az `openMosixanalyzer`, amely az adatokat grafikonon is megjeleníti. Kézzel beavatkozhatunk a programok áthelyezésébe az `openMosixproc` program segítségével. A gépeink manuális adminisztrálásában pedig az `openMosixview` nyújt segítséget. Az `openMosixmigmon` grafikusan jeleníti meg egy adott gépen futó összes programot és azt, hogy melyik gépen futnak ezek éppen.

3. Összefoglalás

A fent ismertetett szoftver előnyei közé tartozik, hogy nincs szükség

párhuzamos programozás használatára. Több platformot is támogat, például a 64 bites AMD és Intel processzorokat is. Telepítésének egyszerűsége is mellette szólhat, illetve kipróbálása nagyon egyszerű a ClusterKnoppix használatával. Ez egy olyan „live-CD”, amelyről ha rendszert hívunk, akkor minden telepítés nélkül egy teljes értékű Linux indul. Több gépen ezt megismételve a gépek automatikusan fürtbe kapcsolódnak, hála az `autodiscovery` programnak. Az openMosix összességében egy nagy perspektívákat nyújtó szoftver, amelyet az SSI technológia fejlődésével egyre több helyen lehet majd eredménnyel használni.