Alkalmazások teljesítményének növelése a folyamatok szinkronizálásával HPC rendszereken

Hahó, fürtcsomópont, most ne dolgozz azon a karbantartási munkán – mindannyian arra várunk, hogy befejezd az MPI munka rád eső részét! Ismerjünk meg egy hatékony fürtkezeléshez használható ütemezési rendszabály.

Azt várhatnánk, hogy ha megduplázzuk az alkalmazásra jutó számítási erőt az alkalmazás teljesítménye is duplájára nő vagy a futásidő felére csökken. Sajnos azonban a *HPC* felhasználók jól tudják ez bizony távol áll az igazságtól, ugyanis az aktuális hatékony teljesítmény akár a rendszer elméleti csúcsteljesítményének 5%-ára is visszaeshet. A *HPC* kutatók és alkalmazás fejlesztők rengeteg időt fordítottak és fordítanak ma is arra, hogy megtalálják az ilyen teljesítményvesztés forrását és feltornásszák az alkalmazás teljesítményét. Amikor elkezdtünk a *Cray XD1* rendszeren dolgozni, magunk is csatlakoztunk a problémával foglalkozó kutatók sorába. Cikkünk arról szól, hogyan tanultunk az előttünk járóktól, és hogyan építettünk erre a tudásra kifejlesztve egy Linux-ütemező alapú megoldást, amely lényegesen megemelheti a *Linux HPC* rendszerek alkalmazásainak teljesítményét.

A legtöbb kutató a *HPC* alkalmazások szerkezetére koncentrált. Különféle kutatócsoportok próbálták növelni a gyorstárazás hatékonyságát, keresték a processzorközi kommunikáció csökkentésének lehetőségeit és vizsgáltak hasonló elemeket, de egyik stratégia sem hozott néhány százaléknál komolyabb javulást. Egy másik kutatási terület azonban sokkal ígéretesebbnek tűnik. Ha megértjük milyen kapcsolat van a *HPC* alkalmazás és a rendszer háttér folyamatai között, megpróbálhatjuk megváltoztatni ezt a kölcsönhatást megnövelve a teljesítményt.

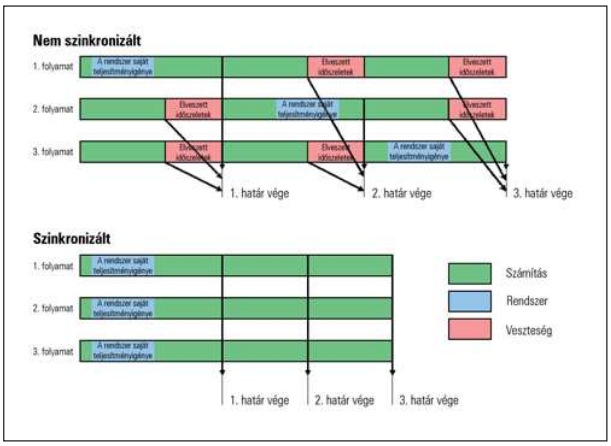
Hova tűnik a teljesítmény?

Ez irányú kutatásaikat ismertető ötletadó írásukban *Petrini*, *Kerbyson* és *Paking* a *Los Alamos National Laboratory* kutatói (lásd a hálózati forrásokat) azt állítják, az alkalmazások teljesítményét egy általuk „zajnak" nevezett tényező okozza – nevezetesen a nagy több ágon futó *MPI* munkafolyamatok és a háttérfolyamatok közötti kapcsolat. Megfigyelték, hogy a karbantartási feladatok, azaz a zaj miatt egyes processzorok nem tudták elérni az *MPI* korlátot (szinkronizálási pontok az alkalmazásban) emiatt az összes folyamatnak rájuk kellett várakozni, míg a processzor végre befejezte a karbantartást. Ez az összes többi processzoron elpazarolt ciklusokhoz vezetett.

Az *1. ábra* felső része ezt a kapcsolatot és az általa okozott teljesítményvesztést mutatja be. A bemutatott folyamatok egy párhuzamos munka részei, valamennyi külön processzoron fut, melyek időnként az *MPI* korlátok segítségével szinkronizálnak. A számítás első felében az 1. folyamat késlekedett, mivel a csomópont ütemezője leállította a folyamat végrehajtását, hogy a minden *Linux* vagy *UNIX* csomóponton megtalálható szokásos háttérfolyamatok egyikét futtassa. A 2. és 3. csomópont szintén késlekedik. E minta megismétlődése jelentősen csökkentheti alkalmazás teljesítményét. A hatás nagyságrendje a korlátok felbukkanásának sűrűségétől és a processzorok számától függ.

Petrini és kollégái ezt a teljesítményvesztést egy az összenyomható fluidumok *Euler* típusú hidrodinamikai viselkedését leíró kód, a *SAGE* futtatásakor figyelték meg *ASCI Q* nevű *HPC* rendszerükön. Az *ASCI Q* egy 2,048 darab *HP ES45* csomópontból álló fürt, ahol minden csomópont maga is négy utas *SMP* rendszer. *Petrini* és társai megfigyelték, hogy amennyiben több mint 256 csomópontot bevontak a számításba, jobb teljesítményt értek el a *SAGE*-el, ha az *SMP* rendszereken a négyből csak három processzort futtattak. Feltételezték, hogy az eltérést a zaj okozza, majd elképzelésüket bizonyították a zajforrások egy részének kiküszöbölésével, amely érzékelhető teljesítménynövekedést okozott.

A kutatások rámutatnak, hogy a folyamat szinkronizálás hiánya és várakozási idő a bűnös, amely az egyébként finoman hangolt és erősen párhuzamosított *HPC* alkalmazások potenciális teljesítményének akár 50%-át (vagy még többet) is elrabolhatja. Sajnos az ilyen tolvajlás kiküszöbölése nem éppen kézenfekvő. A bűnös azonosítására *Petrini* és társai által alkalmazott módszer – a rendszer szabadságfokainak korlátozása a karbantartási munkák végzése terén – a legtöbb *HPC* alkalmazásban nem igazán járható út. A legtöbb *HPC* tulajdonos számára nem igazán kívánatos eljárás, hogy a rendszer processzorainak egynegyedét karbantartási munkák kössék le. Ráadásul a sok háttérfolyamatot nem lehet eltávolítani, így ezzel a megközelítéssel elérhető megtakarítás erősen korlátozott.



1. ábra Példa folyamatok szinkronizált és aszinkron végrehajtására

A hiányzó teljesítmény visszaszerzése

Ha rászánjuk magunkat egy új nagy teljesítményű számítógép összeállítására, valahogy ki kell találnunk, hogyan akadályozhatjuk meg ezt a teljesítménytolvajlást. Kidolgoztunk egy új eljárást, amely a ***Linux*** ütemező segítségével próbálja meg szinkronizálni az ***MPI*** és a karbantartási munkafolyamatok ütemezését. Korábbi munkáink és kutatásaink azt sugallják, hogy az új szinkronizált ütemezési megközelítés akár 50% vagy még komolyabb teljesítménynövekedést is okozhat egyes 32 vagy több processzoron futó, finom hangolt, párhuzamosított alkalmazásnál.

Szinkronizált ütemezési rendszabályok bevezetése

Elképzelésünk alapja egy új ***Linux*** ütemezési rendszabály bevezetése volt. A kívánt előnyök elérése érdekében a rendszabálynak a ***Linux HPC*** rendszer összes gépén szinkronizálnia kell az ütemezőket biztosítva, hogy az ***MPI*** folyamatok az összes folyamattal párhuzamosan futnak ugyanakkor a ***Linux*** karbantartási folyamatok is végrehajtódnak. Így az ütemezőnek valamilyen módon meg kell tudnia valósítani az 1. ábrán bemutatott globális szinkronizálást. A globális szinkron elérése érdekében a kommunikációért felelős processzorhoz terveztünk egy kiegészítést amely az összes feldolgozó csomóponton szinkronizálja az órát. Az új ***Linux*** ütemező rendszabály 128 hellyel rendelkező ütemezési keretet szab meg amelyből 120 az alkalmazás végrehajtására, nyolc pedig a karbantartási munkákra van fenntartva. A különböző processzorok ütemezői a globálisan szinkronizált órához igazíthatják saját ütemező munkájukat, amely a rendszer csomópontjai közt legfeljebb mikroszekundum alatti eltéréseket jelenthet. Bármely időpillanatban az összes processzor vagy a ***HPC*** alkalmazást futtat vagy a karbantartási folyamatokat végez (az 1. ábra alsó fele).
A folyamatszinkronizálás ilyen megközelítése nagy számú processzorra is jól skálázható, hiszen az ütemezési döntést minden csomópont maga végzi el. Az eljárás a korlátokra várakozó ***CPU*** ciklusok megszüntetésével jelentősen megnövelheti a megnyirbált teljesítményt.
A szinkronizált ütemezőt új rendszabályként készítettük el a ***Linux*** kernel ütemezőjéhez már korábban elkészült három létező rendszabály kiterjesztéseként. A ***Linux*** ütemező akkor lép működésbe ha a végrehajtott folyamat elakad, önként átadja a ***CPU***-t, illetve ha a processzor megszakítást érzékel vagy a 10 milliszekundumos időszelet lejár. Az ütemező a következő futtatandó folyamatot az ütemezési rendszabály alapján választja ki a folyamat jellege és fontossága alapján. Az új ütemezési rendszabállyal felvértezett ***Linux*** két valós idejű és a két hagyományos időosztásos folyamatokat támogató ütemezési rendszabály közül választhat. Ezek fontossági sorrendben a következők:

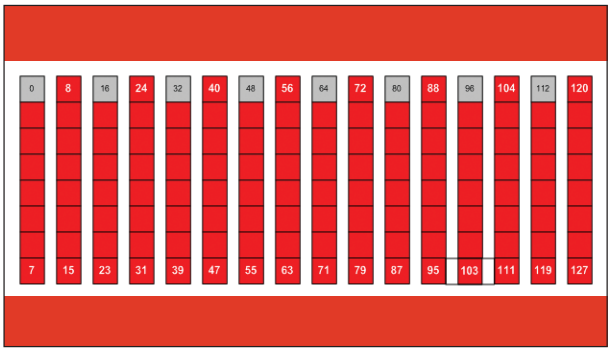
1. ***FIFO (első be, első ki)***: A ***FIFO*** jelölésű folyamat addig fut, míg el nem ereszti a ***CPU***-t. Ezt a prioritást csak rövid ideig használjuk a valós idejű rendszerfolyamatoknál. A ***FIFO*** folyamatok mindig elsőként futnak le.

2. ***Körbejáró (Round robin)***: Az ilyen rendszabályt használó folyamatok sorjában megkapják a 10 milliszekundumos időszeletüket. Valós idejű feldolgozáshoz használatos.

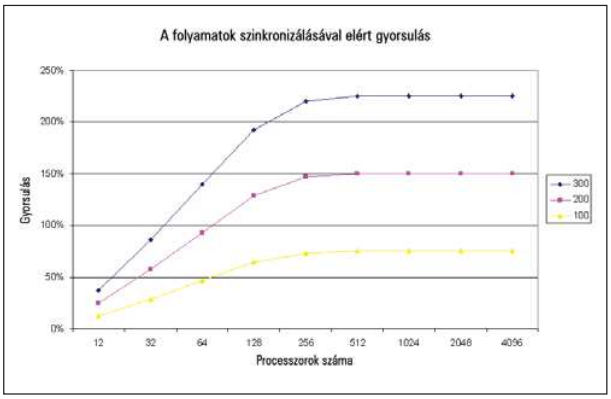
3. ***Szinkronizált***: A szinkronizált rendszabályt azért adtuk a rendszerhez, hogy a kötegelt többprocesszoros munkánál lehetővé tegyük a szinkronizált feldolgozást. A terheléskezelő rendszer minden folyamatot ezzel a rendszabállyal jelöl meg indításkor. A folyamatok és gyermekeik kihasználhatják a szinkronizált ütemezés előnyeit.

4. ***Előjogokon alapuló (Priority)***: Az előjogokon alapuló ütemezés nem más mint a ***Linux*** felhasználók számára jól ismert időmegosztási módszer. Azok a folyamatok melyek ezt a rendszabályt alkalmazzák előjogokkal rendelkeznek és előjogaiknak megfelelően kapnak időszeleteket. Lényegében minden felhasználói és rendszer folyamat ez alatt a rendszabály alatt fut. Az ütemező a legmagasabb fontossági szinten álló osztályból választja ki a következő futtatandó feladatot. A ***FIFO*** és körbejáró rendszerfolyamatok következnek legelőször. A szinkronizált végrehajtásra jelölt feladatok pedig előbb következnek mint a hagyományos előjogos ütemezőben megadottak.

Az új szinkronizált ütemező rendszabály létrehoz egy ütemező keretet, amely eldönti mikor lehet végrehajtani a kötegelt és az egyéb felhasználói illetve rendszer folyamatokat. A keretben előre megadott számú hely van, amelyek sorrendben egymás után következnek. Egy időhely10 milliszekundumot jelent, ***Linux*** alatt ugyanis ennyi egy „rendszerpillanat” (tick), ami alatt a időhelyhez rendelt folyamat futhat. Jelenlegi megvalósításunkban 128 időhely van, amelyből 120 a kötegelt munkák végrehajtására szolgál és 8-at tartunk fenn a rendszerfolyamatok részére. Ez utóbbi időhelyek alatt a szinkronizált ütemező jelzi, hogy nincs futtatandó kötegelt folyamat és így a karbantartási munkákhoz használható hagyományos ütemezési rendszabály lép érvénybe. Amennyiben nincs kötegelt munkafolyamat, a ***Cray*** ütemező megkülönböztethetetlen a megszokott ***Linux*** ütemezőtől.



2. ábra Időhelyek (128) nyolc fenntartott időhellyel



3. ábra Folyamatszinkronizálással elérhető elméleti gyorsulás 1.5% Dćmon CPU foglaltság esetén

Az ütemezési keretben található helyek száma állítható, azonban kötelező kettő hatványnak lennie. A többi folyamattal szemben a kötegelt folyamatok részére fenntartott mennyiség szintén beállítható. A 2. *ábra* egy tipikus ütemezési keretet mutat be, ahol a kötegelt munkák részére fenntartott rész vörös, a karbantartási munkák időhelyei pedig szürke színűek.

Az ütemező keret a csomóponton elindított első kötegelt folyamattal egy időben jön létre. Ilyenkor az összes kötegelt időhely ehhez a folyamathoz tartozik. A további kötegelt folyamatok hozzáadása során az időhelyek egyenletesen eloszlanak a folyamatok közt. Ha ***n*** számú kötegelt folyamatot hozunk létre, az első kötegelt folyamat a helyek 120/***n***-ed részét kapja, a második a következő 120/***n*** időszeletet és így tovább. Így a szinkronizált ütemező olyan kötegelt munkákat is ki tud szolgálni a processzorokon amelyek egynél több folyamatot igényelnek.

A kötegelt folyamat a megkapott idő végéig futhat, feltéve, hogy nincs akadályozva és nincs ***CPU***-t átadó rendszerhívás. Amennyiben a kötegelt folyamat átadja a ***CPU***-t, például mert blokkoló rendszerhívást végez, egy másik kötegelt folyamat kerül végrehajtásra. Amennyiben nincs futtatható kötegelt munka, a vezérlés a hagyományos ütemezőhöz kerül ahol a karbantartási feladatok lefuthatnak. Természetesen a kötegelt folyamat visszakapja a ***CPU***-t amint a megszakítás kezelése után kioldódik.

Ütemezési keretek kiosztása a processzorok közt

Ez idáig kizárólag az egy csomóponton végzett kötegelt munka és rendszerfolyamat ütemezéssel foglalkoztunk.

A teljesítmény elorzásának megakadályozásához, az ütemezőnek valamennyi processzort be kell vonnia. Itt egy rendkívül fontos tervezési feladatba botlunk, amely végső soron lehetővé teszi a szinkronizált ütemezést, nevezetesen a globális időszinkronizálás kérdésébe. Megoldásunkban a globális időszinkronizálást a ***HPC*** rendszerbe tervezett kommunikációs processzorok végzik. Ezek a processzorok a kommunikációs feldolgozás terhét veszik le a többi alkalmazás processzorról. A kommunikációs processzorok az egységes időzítés érdekében egyúttal időszinkronizációs feladatokat is ellátnak. A pontos időszinkronizációt azért érhetjük el, mert a kommunikációs processzorok vezérelni tudják a csomagidőzítést és ugrálást, és így bármely két processzor közötti időeltérés kevesebb lesz mint 1 mikroszekundum. Azzal, hogy a szinkronizációs feladatokat a kommunikációs processzorokkal végeztetjük, további előnyökhöz jutunk, hiszen az alkalmazás terhelést kiszolgáló processzorokról leveszünk bizonyos mennyiségű munkát, így több idő marad az alkalmazás kiszolgálására valamint a megszakítások száma is csökken az alkalmazás processzorokon.

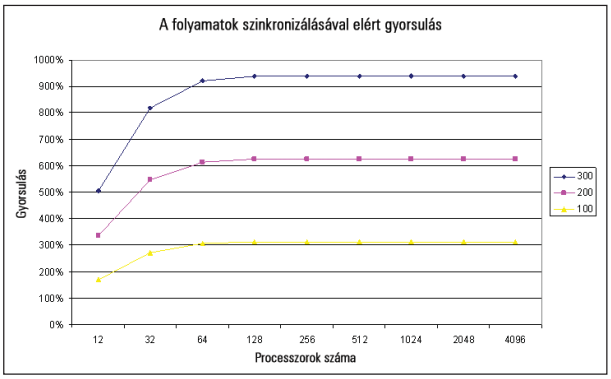
Az idő szinkronizációs protokoll kiegészítő helyeket tartalmaz az időhely kiosztáshoz. A protokoll mester szolga elgondolás szerint működik, ahol az egyik csomópont szolgál idő és időhely információ forrásként az összes többi csomópont pedig a mester csomópont órájához igazodik. A mestertől kapott időszinkronizálási csomagok azonosítják a végrehajtandó időhelyet és az időhely indítása óta eltelt időt, így a teljes ***HPC*** rendszerben precízen megadhatóak az ütemező keretek.

Teljesítmény kérdések

A szinkronizált ütemező a párhuzamos alkalmazások folyamatainak szinkronizált végrehajtását teszi lehetővé. Hogy mekkora teljesítménycsökkenést lehet elkerülni, vagy mekkora potenciális teljesítmény érhető el, az az alkalmazás által használt korlátok és gyűjtőműveletek számától, a rendszer karbantartó folyamatok által elhasznált időmennyiségtől és az alkalmazásban felhasznált processzorok számától függ.

Kutatásaink azt mutatják, hogy a módszerrel jelentős gyorsulást lehet elérni. A 3. és 4. *ábra* a szinkronizált ütemező által elérhető elméleti gyorsulást mutatja a hagyományos előjogos ütemezővel szemben. A 3. ábrán feltételeztük, hogy a háttérfolyamatok a processzor 1.5%-át foglalják le, a 4. ábra szerint pedig a háttérfolyamatok a ***CPU*** 6.25%-át igénylik – ezek egyébként a legtöbb valódi fürtön tapasztalható értékek. A bemutatott görbék másodpercenkénti 100, 200 és 300-as korlátszámnak felelnek meg.

Ahogy a processzorok száma növekszik, a szinkronizált ütemező nyújtotta előnyök aszimptotikusan közeledik a maximum felé. Ez azt a tényt tükrözi, hogy a hagyományos ütemezővel sem csökken akármeddig a teljesítmény. Bizonyos processzorszám elérése után annak a valószínűsége, hogy legalább egy processzor karbantartási munkát végez, közelít a 100%-hoz. További processzorok hozzáadása lényegesen már nem növeli a korlátoknál észlelhető alkalmazás csúszást.



4. ábra Folyamatszinkronizálással elérhető elméleti gyorsulás 6.5% Dćmon CPU foglaltság esetén

Összefoglalás

A *HPC* alkalmazás és a rendszer háttérfolyamatai közötti kölcsönhatásra fokuszálva, a *HPC* kutatók rátaláltak a párhuzamos alkalmazások teljesítménycsökkenésének egyik fő okára. További kutatások e tolvajlás megakadályozásának lehetőségére is fényt derítettek, de ez egyelőre még egyetlen valódi működő megoldás sem létezik. A *Linux* ütemezőjét használó globális folyamat szinkronizálás megszünteti a zaj várakozási időt és jelentős teljesítménynövekedéssel kecsegtet. A *HPC* rendszer egyéb szabályai és az alkalmazáson túllépve úgy hisszük, rábukkantunk egy valódi, komolyan használható megoldásra. Globális óraszinkronizálást alkalmazó *Linux* folyamatszinkronizálással és az egyes csomópontokon futtatott *Linux* rendszerekkel a *Cray* megoldása az összes processzoron képes biztosítani az alkalmazás folyamatainak párhuzamos futását miközben a karbantartási folyamatok kötött időben, szintén párhuzamosan futnak le. Folyamatszinkronizáló megoldásunk megelőzheti a teljesítményvesztést és akár 50%-al is megnövelheti a finom felbontású erősen párhuzamosított alkalmazások teljesítményét a 32 vagy több processzort alkalmazó rendszereken.

Linux Journal 2004. november, 127. szám

Dr. Paul Terry a Cray által 2004 áprilisában felvásárolt, korábban OctigaBay Systems néven ismeretes, Cray Canada vezető műszaki szakembere. Az új számítási szerkezetek technológiai stratégája, aki a cég műszaki elképzeléseinek és vezető helyzetének megalapozásáért felelős.

Amar Shan a Cray termelésirányítási igazgatója, a Cray élvonalbeli műszaki újításaiért és kreatív üzleti megoldásaiért felelős. Több mint 20 éves tapasztalattal rendelkezik a számítási és telekommunikációs ipar termékmenedzsmentje, fejlesztése és architektúra szabályai terén.

Pentti Huttunen a Cray teljesítménymérési szakértője aki a párhuzamos számítási eljárások kidolgozásáért és az alkalmazások optimalizálásért felelős. Munkájának fő célja az, hogy ezek az alkalmazások a Cray különféle gépein a lehető leghatékonyabban fussanak.