Ellenőrzőpont támogatás PVM alkalmazások számára a magyar ClusterGriden[1]

Kovács József, Farkas Zoltán, Marosi Attila

MTA SZTAKI Párhuzamos és Elosztott Rendszerek Laboratórium
1518 Budapest, Pf. 63.
{smith, zfarkas, atisu}@sztaki.hu

Bevezetés

A magyar ClusterGrid futtató környezetben hosszan futó párhuzamos folyamatokból álló alkalmazások nem futtathatók a résztvevő pc-k nappali-éjszakai váltott üzemmódú működése miatt. A probléma megoldására az MTA SZTAKI LPDS laboratóriuma kidolgozott egy olyan szolgáltatás központú ellenőrzőpontozó (checkpoint) rendszert, mely nem igényel támogatást az ütemezőtől. A felsőbb szinten működő bróker mindössze az ellenőrzőpontozás eredményeként létrejött információk mozgatásáért felel.

Az új megoldás lehetővé teszi a ClusterGrid rendszeren népszerű PVM alkalmazások automatikus felfüggesztését a nappali üzemmódra kapcsolás előtt és folytatását az éjszakai (Grid) üzemmód bekapcsolása után. Mindez a felhasználók számára transzparens módon történik, azaz PVM programjukat nem kell módosítani az ellenőrzőpontozó rendszer alkalmazhatósága érdekében. A kidolgozott megoldás a ClusterGrid megbízhatóságát és hibatűrő képességét is nagymértékben növeli. Ha egy klaszter vagy annak egy gépe bármilyen okból kiesik a rendszerből az ellenőrzőpontozó rendszer segítségével az utolsó ellenőrzőponttól folytatható az alkalmazás más erőforrásokra kiosztva azokat a PVM taszkokat, amik eredetileg a meghibásodott erőforráson futottak. A ClusterGrid bróker megfelelő továbbfejlesztésével lehetőség nyílik arra is, hogy a PVM programok dinamikusan, az optimális terhelésnek legjobban megfelelő módon migráljanak a Grid erőforrásai között.

A Cluster Grid rendszer

A Cluster Grid projekt [1] célja összefogni és integrálni klaszterbe kötött PC-ket egy nagy országos méretű griddé. A PC-ket a résztvevő magyar oktatási intézetek nyújtják, a központi infrastruktúrát és a koordinálást pedig az NIIF végzi, amely a Magyar Akadémiai Hálózat üzemeltetője is egyben. Minden tag a klasztert nappal a saját oktatási céljaira használja, többnyire Windows operációs rendszerrel, majd használaton kívüli üzemmód esetén pl. éjszaka vagy hétvége, a gépek linux operációs rendszert futtatnak. Ebben az esetben a gépek oly módon vannak konfigurálva, hogy egy klasztert alkossanak és egyúttal csatlakozzanak a ClusterGrid országos grid infrastruktúrához. A nappali-éjszakai üzemmód lehetővé teszi a számítógépek erőforrásainak hatékony kihasználását, hogy azt a magyar kutatói közösség rendelkezésére bocsáthassa.

Jelenleg mind szekvenciális, mind párhuzamos alkalmazások futtathatóak a rendszeren. Ez utóbbi esetében a PVM (Parallel Virtual Machine) [2] a leginkább támogatott szoftverfejlesztői könyvtár. Az automatikus ellenőrzőpontozás a szekvenciális alkalmazások számára működik, azokon belül is a statikusan linkelt végrehajtandó állományok esetén. Mindezek folyománya, hogy a párhuzamos alkalmazások nem futhatnak tovább, mint 10 óra, mert az üzemmód váltás mindenképpen befejezi az alkalmazás végrehajtását és ellenőrzőpontozási technika nélkül minden addigi feldolgozás elvész. Felhasználói szintű ellenőrzőpontozás ugyan létezik, de annak

[1] A cikkben leírt munkát a következő kutatási grantok támogatták: az IHM 4671/1/2003 projekt, az OTKA T042459 és a Magyar SzuperKlaszter OMFB-00495/2004 projekt

használhatósága erősen korlátozott. Az alkalmazás nem helyezhető át más klaszterekre és magát az alkalmazás speciálisan kell kódolni.

Párhuzamos programok checkpointolásának nehézségei

Az elosztott checkpointoló rendszernek három alapvető problémát kell megoldania. Képesnek kell lennie az alkalmazást alkotó processzek állapotterének, az üzenetközvetítő alrendszerben keringő üzeneteknek és a processzek közötti kapcsolatoknak a lementésére és visszaállítására. Továbbá mindezt oly módon kell tennie, hogy üzenetek ne duplikálódjanak, ne vesszenek el és célba érjenek egy esetleges processz másik futtató egységre történő áthelyezését követően is.

Kommunikáló processzek esetén az alkalmazás konzisztens állapotának lementését nagyban nehezítik a rendszerben bolyongó üzenetek. Az alkalmazás lementése oly módon történik, hogy a processzek végrehajtásának felfüggesztését követően a teljes memóriatérképet és végrehajtási kontextust leolvassuk és fájlba mentjük. Az alkalmazás azonban nem függeszthető fel bármely időpillanatban, hiszen egy megszakított üzenetküldés vagy üzenetátvétel programhibát okozhat akkor, ha adott paraméterekkel az eredeti környezetben megkezdett üzenetküldés már egy átrendezett, új környezetben kerül lefutásra a visszaállítást követően. Ez esetben a környezet a processzt körülvevő kommunikációs kapcsolatot és a partnereket jelenti.

Az üzenetközvetítésen alapuló alkalmazások checkpointolásának alapvető nehézsége, abban rejlik, hogy egy folyamatosan kommunikáló processz halmazban nem tudunk egyértelműen olyan időpontot meghatározni, amikor a processzek nem kommunikálnak és üzenetek sincsenek az üzenetközvetítő alrendszerben. Az üzenetközvetítő réteg változtatására nincs lehetőség, így arra sincs módunk, hogy egy adott időpillanatban esetleg lementsük e réteg belső állapotterét. Tehát valamilyen módon szükség van annak biztosítására, hogy az állapottér lementésekor ne legyenek keringő üzenetek a rendszerben, ellenkező esetben üzenetvesztés vagy duplikálódás történhet.

Megoldandó továbbá az üzenetek célba érkezésének biztosítása is. Tegyük fel, hogy egy processz üzenetet küld egy másiknak, és éppen az átvétel előtt mentjük le az alkalmazás állapotterét. Lementést követően a fogadó processzt áthelyezzük egy másik gépre, majd innen folytatjuk a végrehajtást. Nyilvánvaló, hogy az üzenet nem fog célba érni, hiszen a célállomáson már nem létezik a címzettként megjelölt processz. Az üzenetközvetítő alrendszert pedig nincs lehetőség oly módon megváltoztatni, hogy az ilyen eseteket intelligensen lekezelje, az üzenet újrapostázásával.

Már csak azért sem, mert ha erre fel lenne készülve a kommunikációs alréteg, akkor sem lenne működőképes egy újabb akadály miatt. Ez pedig nem más, mint a processzek kommunikációs azonosítóinak megváltozása egy újbóli belépés esetén. Tehát az eredeti üzenet által tartalmazott címzett többé már nem létezik, attól a pillanattól, hogy kilépett a kommunikációs rétegből. Márpedig ez elkerülhetetlen, ha migrálni akarjuk, vagy esetleg teljesen újra szeretnénk indítani az alkalmazást.

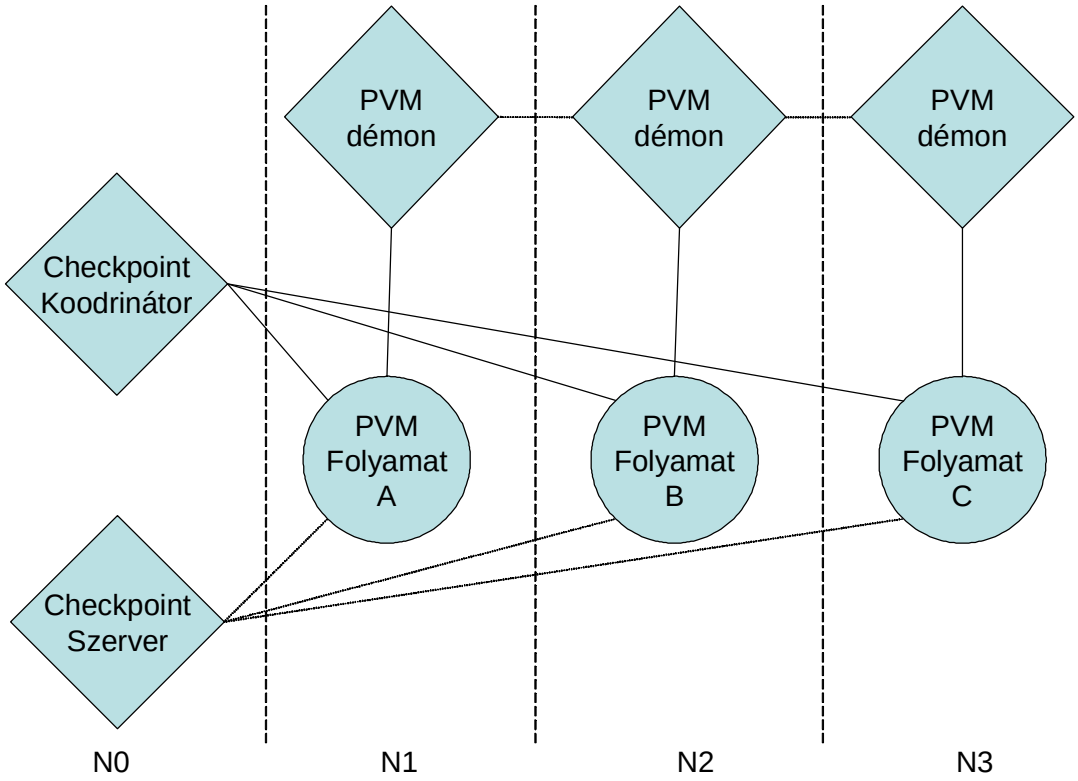
További nehézséget okoz a szoftverkörnyezet által felállított igények, követelmények. A jelenleg működő clustergrid környezetben condor [3] pool-ok működnek. Ebben az esetben a condor saját, módosított üzenetközvetítő alrendszert használ. Konkrétan ez azt jelenti, hogy a pvm a condor alatt egy speciális módosított verzió, tehát nincs lehetőség a pvm démonok oly módon való módosítására, amely segítséget nyújthatna az ellenőrzőpontozás megvalósításához.

A PVM alkalmazás és környezetének felépítése

A CulsterGrid checkpointoló rendszerében a pvm alkalmazás checkpointolása a következő komponensek együttműködésével valósul meg:

- PVM alkalmazás
  A felhasználó által írt tetszőleges pvm alkalmazás.
- PVM démonok
  Módosítatlan üzenetközvetítő háttér foylamatok, melyek a pvm virtuális gépet alkotják együttesen és a folyamatok közötti információcserét végzik.

- Checkpoint Koordinátor
  Egy háttérben futó folyamat, mely a PVM alkalmazást alkotó folyamatok lementését és visszaállítását koordinálja.
- Checkpoint Profiling Könyvtár
  Az alkalmazáshoz szerkeszthető könyvtár, mely a pvm függvények viselkedését képes befolyásolni.
- Checkpointer Eszköz
  Az alkalmazáshoz szerkeszthető könyvtár, mely egyetlen folyamat állapotterét képes lementeni vagy checkpoint információ alapján visszaállítani.
- Checkpoint Szerver
  Egy háttérben futó folyamat, mely a checkpoint információt gyűjti össze a folyamatoktól, illetve kérés esetén visszaszolgáltatja.



1. ábra Az alkalmazás és kapcsolata a külvilággal

A Checkpoint koordinátor

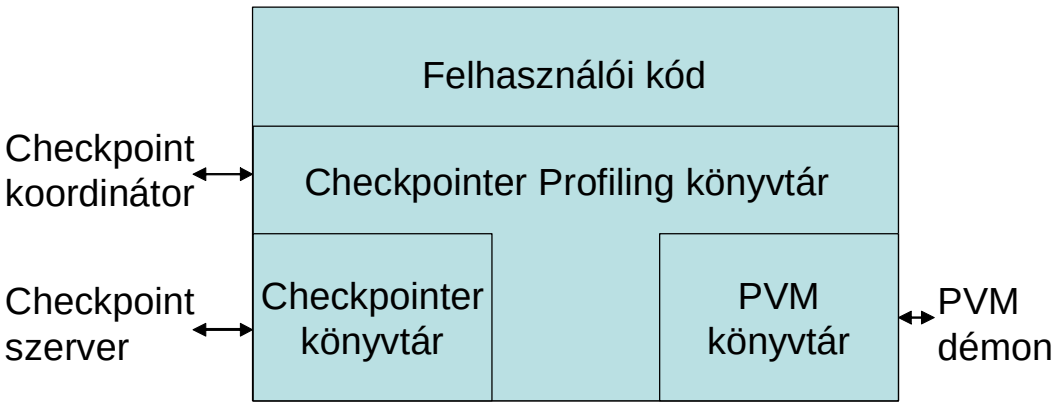
A koordinátor egy a háttérben folyamatosan futó kiszolgáló folyamat, mely a pvm alkalmazások checkpointolhatóságához ad segítséget. A koordinátor a következő feladatokat látja el:

- újonnan indított folyamatok számára
  - feléledési üzemmódot (normál vagy checkpointból) határoz meg
  - a checkpoint információ elérését adja meg
  - a már futó folyamatok azonosítóinak listáját szolgáltatja
- alkalmazás folyamatainak állapot lementésekor
  - ütemezi az állapotlementéshez szükséges lépéseket
  - az üzenet szinkronizációban résztvevő folyamatokat azonosítja
  - elkészíti az alkalmazás állapotának leírását
- alkalmazás állapotának visszaállításakor

    - a korábban elmentett alkalmazás állapotának leírását dolgozza fel
    - folyamat(ok) indítására utasít, mellyel felépíti az alkalmazást
- az alkalmazás futása során
    - nyomon követi az alkalmazás folyamatait
    - üzenetküldés címzettjének hiánya esetén azonosító kiszolgálása
    - hibás állapotokat ismer fel és kezeli le
    - kilépő folyamatokat észleli

A Checkpoint profiling könyvtár

A checkpointoló rendszer egyik legfontosabb alappillére a pvm alkalmazás viselkedésének módosítását végző profiler könyvtár. Ez egy olyan program réteg, mely az alkalmazás és a pvm kommunikációs alkönyvtár közé ékelődik be, módosítja a pvm rutinok viselkedését oly módon, hogy az checkpointolhatóvá váljon. Az alkalmazás rétegződését a 2. ábra szemlélteti.



2. ábra A PVM alkalmazás egy folyamatának rétegződése és kapcsolatai

A Checkpointer eszköz/könyvtár és Checkpoint szerver

A checkpointoló rendszeren belül a folyamatok memóriatartományának tartalmát ténylegesen elmentő eszköz egy beépített statikus könyvtár. Ez egy szignál kezelőt inicializál a procesz indulásakor, majd a szignál érkezésekor átveszi a vezérlést és megkezdi a folyamat állapotterének, memóriatartományának lementését. Mindezek előtt természetesen a folyamatot elő kell készíteni az üzenetek és kapcsolatok védelme érdekében, amit a checkpoint profiling könyvtár végez. Amikor a checkpointer eszköz másolatot készít a folyamat állapotteréről, azt a szervernek küldi el, mely file-ba helyezi az információt. Ez a checkpoint szerver rendszerint a koordinátorral megegyező gépen fut. A checkpointer eszköz és szerver jelenleg a Vic Zandy által fejlesztett ckpt [4] nevű eszköz.

Az alkalmazás életciklusa

Egy alkalmazás teljes életciklusa alatt alapvetően 4 fő eseményt definiálhatunk, melynek során a checkpointoló rendszer komponensei aktív tevékenységet végeznek.

1. Folyamat létrejötte
    a. Normál indítással
    b. Checkpointból történő feléledés
2. Állapot lementés
3. Folyamat (sikeres vagy hibás) leállása

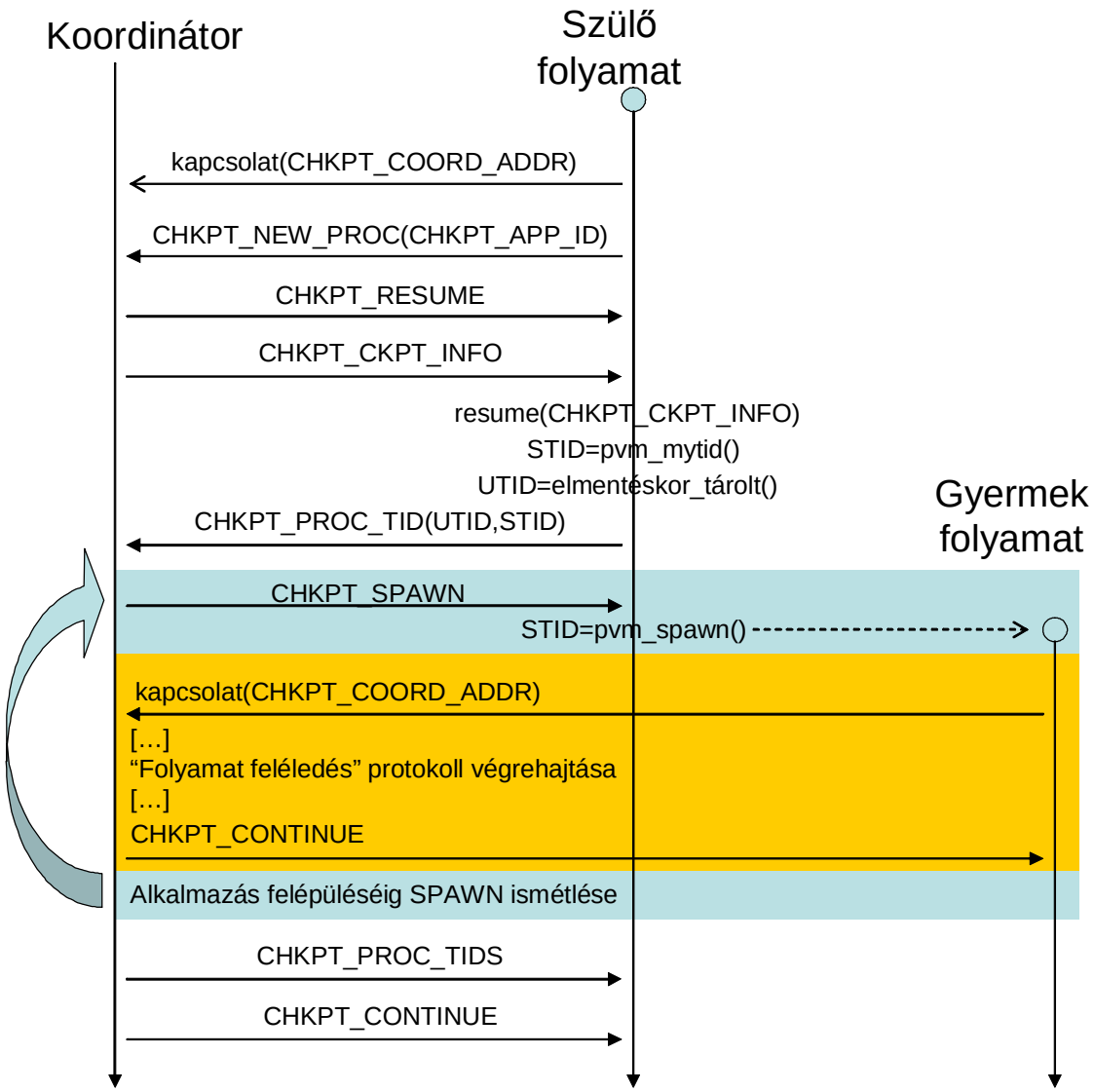
A 4 fő működési üzemmód közül a továbbiakban a feléledés és lementés életciklusokat ismertetjük, melyek a legérdekesebb és egyben legösszetettebb eljárások. A következőkben bemutatásra kerülő protokollokat a checkpointer profiling könyvtár valósítja meg a PVM folyamatban a protokoll másik résztvevője a koordinátor.

Checkpointból történő feléledés

Checkpointból történő feléledés azt jelenti, mikor az alkalmazás folyamatai indulásukat követően egy előzőleg elmentett állapotba állítódnak vissza.

Minden folyamat újraindításakor a következő lépések hajtódnak végre:

1. Az elindított folyamat a CHKPT_COORD_ADDR környezeti változó értéke alapján felveszi a kapcsolatot a koordinátorral.
2. Az elindított folyamat a CHKPT_RESUME környezeti változó jelenléte alapján dönti el, hogy normál indítás vagy checkpointból való visszaállítás szükséges. Normál futás RUN, a checkpointból való visszaállítás RESUME üzemmód. A továbbiak RESUME esetre vonatkoznak, azaz a változó be van állítva.



3. ábra Folyamatok egymás utáni feléledése

3. Az elindított folyamat bejelentkezik a koordinátorhoz, elküldi a CHKPT_APPID azonosítóját, mely segítségével azonosítja magát. Opcionálisan paraméterként szerepel a RESUME üzemmód megjelölése is, majd várakozik a koordinátortól a megfelelő üzemmódra.

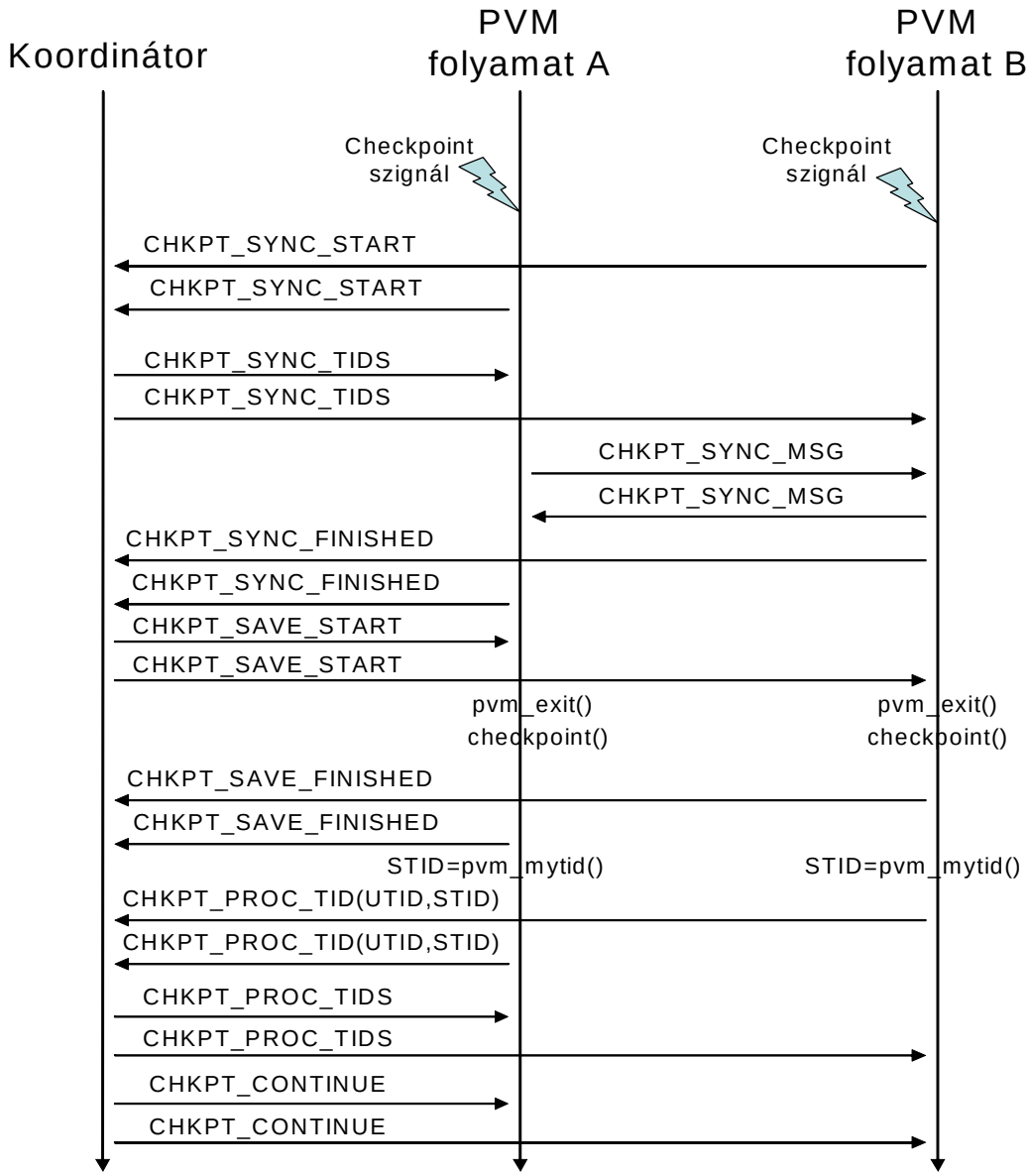
4. A koordinátor ellenőrzi a RESUME üzemmódú indítás feltételeit, majd CHKPT_RESUME üzenettel utasítja az indított folyamatot a RESUME üzemmódú futásra.
5. A koordinátor elküldi az indított folyamathoz tartozó checkpoint információ azonosításához és eléréséhez szükséges paramétereket (CKPT_SERVER, CKPT_ID), az indított folyamat biztonságos módon eltárolja azt.
6. Az indított folyamat visszaállítja magát a checkpoint file-ból.
7. Az indított folyamat a biztonságos módon eltárolt paramétereket – melyek a checkpointoló eszköz működéséhez szükségesek – beállítja a visszaállítás előtti állapotra.
8. Az indított folyamat csatlakozik a PVM-hez. A kapott PVM_TID érték ezentúl a SYSTEM_TID, a USER_TID pedig az első indítás alkalmával eltárolt lesz.
9. A folyamat elküldi a USER_TID, SYSTEM_TID értékeit, majd újabb koordinátor üzenetre vár. A USER_TID érték ütközésének ellenőrzésére nincs szükség, mert az lementéskor bizonyíthatóan csak duplikációmentes értékeket tartalmazott.
10. Ha első bejelentkező folyamatról van szó, a koordinátor elküldi a folyamat számára az újabb folyamat indításához szükséges paramétereket a CHKPT_SPAWN üzenetben.
11. A koordinátor mindaddig ismétli az 10. lépést, amíg az alkalmazás összes folyamata fel nem éled és el nem jut erre a pontra.
12. A koordinátor CHKPT_PROC_TIDS üzenetben elküldi minden indított folyamatnak az aktuális USER_TID, SYSTEM_TID párosításokat.
13. A koordinátor CHKPT_CONTINUE üzenettel jelzi minden folyamatnak, hogy továbbfuthat.
14. A folyamatok megkezdik, azaz egy korábbi állapottól folytatják futásukat.

Alkalmazást alkotó folyamatok állapotterének lementése

Az alkalmazás állapotterének lementését az egyes folyamatok állapotának és üzeneteinek szinkronizációja előzi meg. Állapotlementés előtt a pvm virtuális gépből kilép a folyamat. Ezen fázisok megvalósításához a következő lépések szükségesek:

1. Az alkalmazást alkotó minden folyamatnak kapnia kell a checkpointolás megkezdését jelző unix szignált.
2. Minden folyamat a checkpoint szignál megérkezését követően CHKPT_SYNC_START üzenettel jelzi a koordinátornak, hogy készen áll az állapot lementési eljárás megkezdésére, majd válaszra vár.
3. A koordinátor megvárja, míg minden folyamattól meg nem érkezik az értesítés. A folyamatok számát az alkalmazáshoz tartozó aktív kapcsolatai alapján tudja meghatározni a koordinátor.
4. Ha minden folyamattól megérkezett a CHKPT_SYNC_START üzenet, a koordinátor CHKPT_SYNC_TIDS üzenetben elküldi minden folyamatnak az aktuális USER_TID, SYSTEM_TID párosításokat, melyek az első indítás idejében illetve jelenleg érvényben lévő PVM_TID értékek. Ezután a koordinátor a folyamatokra vár.
5. A CHKPT_SYNC_TIDS hatására a folyamatok elvégzik az üzenet szinkronizációt.
6. Üzenetszinkronizáció sikeres befejeztét minden folyamat jelenti a koordinátornak egy CHKPT_ SYNC_FINISHED üzenettel.
7. A koordinátor CHKPT_SAVE_START üzenettel engedélyezi az állapot lementés előkészítését és elvégzését.
8. A folyamat a CHKPT_SAVE_START üzenet hatására elhagyja a PVM-et.
9. A folyamat állapotterének lementése megtörténik.
10. A lementés után a folyamat CHKPT_SAVE_FINISHED üzenettel jelzi a lementés sikerességét.
11. A folyamat belép a PVM-be, majd a koordinátornak küldött CHKPT_PROC_TID üzenettel jelzi futásra való felkészültségét. A USER_TID, SYSTEM_TID értékeket is elküldi paraméterként.
12. Minden folyamat CHKPT_PROC_TIDS üzenet megérkezését követően eltárolja az üzenetben lévő folyamatok PVM_TID értékeit.

13. A koordinátor a CHKPT_CONTINUE üzenettel engedélyezi az alkalmazás összes folyamatának a végrehajtás folytatását.
14. A folyamatok tovább futnak.



4. ábra Alkalmazás állapotának lementése és a futás folytatása

ClusterGrid illesztés

A fent vázolt protokoll alapján működő (ellenörzőpontozó) checkpointoló rendszer egy általános PVM alkalmazások számára kifejlesztett megoldás, mely segítségével az alkalmazások tetszőlegesen migrálhatók a node-ok, sőt a klaszterek között is. A rendszer működéséhez szükséges külső támogatások a következőek:

- Az elindított PVM alkalmazás fordítását a checkpointer rendszer által megadott kapcsolókkal, paraméterekkel kell fordítani
- Az alkalmazás indításakor a checkpoint rendszer számára szükséges környezeti változókat kell beállítani

- A checkpointolás megindításához a PVM alkalmazás összes folyamatának meg kell kapnia a checkpoint szignált
- A migrációhoz szükséges checkpoint információk mozgatásáról gondoskodnia kell

A fent definiált igényeket a ClusterGrid környezetben az egyes klasztereken futó lokális ütemező elégíti ki.

Összefoglalás

A cikkben ismertetett checkpointoló és migráló rendszer általános PVM alkalmazások számára lett megtervezve. Képes egy általános PVM alkalmazás állapotterét lementeni, majd egy vagy az összes folyamatát egy másik gépen vagy klaszteren újraindítani az elmentett állapotból. Mindezt egy háttérben futó koordinátor ütemezi, felügyeli. A checkpointolás kezdetét egy a PVM alkalmazás összes folyamatának küldött szignál jelzi. A végeredmény file-okban tárolt állapotinformáció. A migráció a file-ok egy másik klaszterre való átmozgatásával és újrasubmittálásával érhető el. Az alkalmazás folyamatainak feléledésekor a helyi checkpoint koordinátor elvégzi a szükséges újraélesztést a megadott állapotinformációk alapján.

Hivatkozások

[1] ClusterGrid projekt http://www.clustergrid.iif.hu/
[2] PVM honlapja: http://www.epm.ornl.gov/pvm
[3] Condor honlapja: http://www.cs.wisc.edu/condor
[4] Ckpt honlapja: http://www.cs.wisc.edu/~zandy/ckpt