Konzisztens állapottér átvitel üzenetközvetítéssel kommunikáló párhuzamos algoritmusok számára Grid környezetben

Phd értekezés tézisei

Készítette:

Kovács József
tudományos munkatárs
MTA SZTAKI

Témavezetők:
Dr. Kacsuk Péter (MTA SZTAKI)
Dr. Kovács László (ME)

Doktori iskola vezetője:
Dr. Tóth Tibor

Magyar Tudományos Akadémia,
Számítástechnikai és Automatizálási Kutató Intézet
(MTA SZTAKI)

Hatvany József Informatikai Tudományok Doktori Iskola,
Miskolci Egyetem
(ME)

Budapest, 2008

Bíráló bizottság tagjai

Elnök:

Dr. Arató Péter, az MTA levelező tagja (BME-VIK)

Titkár:

Dr. Czap László, PhD (ME GÉK)

Tagok:

Dr. Stefán Péter, PhD (NIIF Intézet)

Dr. Juhász Zoltán, PhD (Pannon Egyetem, Veszprém)

Dr. Erdélyi Ferenc, CSc (ME, nyugalmazott egyetemi docens)

Dr. Sima Dezső, DSc (BMF NJIK)

Opponensek:

Dr. Kovács Szilveszter, PhD (ME GÉK)

Dr. Szeberényi Imre, PhD (BME-VIK)

I. Bevezetés

Számos tudományterületen jelennek meg egyre nagyobb számítási kapacitást igénylő feladatok, melyeket egyetlen nagy teljesítményű számítógéppel már nem lenne gazdaságos vagy nem lenne lehetséges ésszerű időn belül feldolgozni. A nagyobb számítási kapacitás eléréséhez mind több és több számítógép erőforrásainak egyidejű kihasználását, egységbe foglalását kell megvalósítani. Ez a célja napjainkban az egyre nagyobb tért hódító Grid [A] kutatásnak. A Grid, mint infrastruktúra célja a földrajzilag szétszórtan elhelyezkedő erőforrások megosztott, közös használata ellenőrzött, biztonságos és kölcsönösen előnyös módon. Építőelemei lehetnek egyedi, illetve szorosan együttműködő csoportba, ún. fürtbe vagy klaszterbe (cluster) szervezett számítógépek. Azon infrastruktúrára, melyben az építőkövek kizárólag klaszterek, általában a KlaszterGrid (clustergrid) elnevezés használatos a tudományos szakirodalomban.

Az egyre nagyobb számítási kapacitás kiaknázására leginkább elosztott módon és több gépen, egyidejűleg futó algoritmusokkal valósítják meg. Az alkalmazások futás közben üzenetküldés segítségével cserélnek információt, vagy részeredményeket egymás között. Az ilyen típusú algoritmus illetve alkalmazás egyik tipikus futtató környezete a klaszter, mely egyrészt az általa nyújtott erőforrások hatékony ütemezését végzi, másrészt háttérben futó szolgáltatások segítségével a gépek között zajló kommunikációt támogatja. Előbbit az ún. feladatütemező (pl. Condor [B][C]) végzi, utóbbi célra pedig valamilyen kommunikációs könyvtár (pl. PVM [D][E] vagy MPI [F][G]) használható.

Egy Grid környezetben a számítási kapacitást igénylő alkalmazásokat az ún. bróker rendeli az őket futtatni képes erőforrásokhoz vagy klaszterekhez. Ha egy klaszteren futó alkalmazás valamilyen oknál fogva (erőforrás adminisztráció, erőforrás túlterhelés, erőforrás hiba) nem képes tovább futni, szükségessé válik az alkalmazás másik erőforrásra történő áthelyezése (migráció). Hagyományos esetben az alkalmazás teljesen újraindul a számára újonnan kiválasztott klaszteren, mely rendkívül nagy számítási kapacitás veszteséget okozhat a rendszerben. Ennek kiküszöbölésére léteznek ún. állapot-visszaállítási (rollback-recovery) technikák [H], melyek képesek az alkalmazás futását egy (korábban elmentett) állapotból folytatni.

A szakirodalomban számos technikát lelhetünk fel az üzenetközvetítéssel kommunikáló párhuzamos vagy elosztott algoritmusok állapotának elmentésére és futásának az elmentett állapotból történő folytatására, melyek közül a legelterjedtebb az ún. *ellenőrzőpontozás* [I] (checkpointing). Az ellenőrzőpontozás témakörben a számos nemzetközi kutatás eredményeképpen számos módszer került kidolgozásra, melyek különböző szempontok (megvalósítási szint, koordináció típusa, stb.) alapján osztályozhatóak.

Az ellenőrzőpontozási technikák használatának fontossága megkérdőjelezhetetlen hosszan futó párhuzamos alkalmazások esetén [J], ennek ellenére elterjedésük Grid környezetben még nem valósult meg. Az egyik akadály, hogy a jelenleg elérhető eszközök rendszerint olyan szolgáltatásokon alapulnak, melyek nem feltétlenül találhatóak meg az egyes klasztereken. Mivel az egyes klaszterek különböző szoftverháttérrel rendelkezhetnek, olyan ellenőrzőpontozási módszerre van szükség, mely nem függ külső szolgáltatástól (transzparencia).

Kutatási tevékenységem során felhasználásra került az MTA SZTAKI által fejlesztett P-GRADE párhuzamos programozási környezet. A P-GRADE [1][K] integrált környezet grafikus megoldást nyújt a párhuzamos alkalmazások fejlesztéséhez és végrehajtásához klasztereken, szuperszámítógépeken és Grid rendszereken. A P-GRADE meglévő szekvenciális programok újratervezését gyorsítja fel, melyhez egy hierarchikus tervezési eszközt, egy hibrid grafikus nyelvet, a GRAPNEL-t [L], hibakeresést és –javítást [2], tesztelést, on-line monitorozást, teljesítményanalízist [1], valamint vizualizációs lehetőségeket nyújt.

A P-GRADE eszközt számos felsőoktatási intézményben használták fel a párhuzamos algoritmusok és rendszerek témakör oktatásához, továbbá a prototípus piacra való bevezetése – az akadémiai szférából való továbblépésként – jelenleg egy alakuló angol-magyar közös vállalat céljai között szerepel.

A bemutatott munka megkísérel túllépni az említett korlátokon egy transzparens ellenőrzőpontozó és migrációs módszer kidolgozásával, az azt kielégítő absztrakt modell definiálásával, illetve konkrét megoldások bemutatásával.

II. Alkalmazott módszerek

Kutatásom során elsődleges célom volt az ellenőrzőpontozási technika integrálása KlaszterGrid környezetbe. Ehhez első lépésként meghatároztam az általam kiindulási pontként tekintendő KlaszterGrid fontosabb működésbeli és felépítésbeli követelményeit elsősorban szakirodalmi feldolgozással és a jelenleg működő Grid rendszerek áttekintésével. Az infrastruktúra tanulmányozását követően irodalomkutatást végeztem a létező ellenőrzőpontozási technikák, módszerek [H][I][J] feltérképezése érdekében, majd hasonlóan jártam el a már meglévő konkrét rendszerek analízisekor [M][N][O][P][Q][R][S]. A megvizsgált eszközöket különböző szempontok szerint rendszereztem [3].

A követelmények azonosítását követően, egy formális modell megalkotását tűztem ki célul. A formális modell leírása céljából az elérhető módszerek közül az ASM [T][U][V] (Abstract State Machine) módszert választottam. Az ASM tetszőleges absztrakciós szinten képes rendkívül rugalmasan kifejezni és kezelni a modellt alkotó elképzeléseit. Az általam készített modelleket ennek megfelelően ASM rendszerben készítettem, majd modellfinomítási eljárást alkalmaztam és a kidolgozott modellek közötti finomítások helyességét igazoltam ASM-el.

Az ASM formális keretrendszert számos ipari és tudományos projekt használta nagyobb rendszerek tervezésére, analízisére. Sikeresen alkalmazták többek között a Prolog [W], Occam [X] fordítók és Java Virtuális Gép [Y] végrehajtásának ellenőrzésére, továbbá a Microsoft is alkalmazza szoftver rendszerek tervezéséhez és analíziséhez [Z].

A módszer kidolgozását követően a P-GRADE rendszer belső felépítésének analízisére volt szükség a megfelelő adaptáció megtervezéséhez. Mindezt egyrészt a szakirodalom feldolgozásával, másrészt a rendszer tesztelésével, belső vizsgálatával értem el. Ugyanezt alkalmaztam a TotalCheckpoint [4] ellenőrzőpontozó rendszer megalkotásánál is, melyet az egyik legelterjedtebb üzenetközvetítő réteg, a PVM behatóbb tanulmányozása előzött meg. Az egyes tervek elkészítésekor szintén alkalmaztam a már korábban említett ASM absztrakt modell-készítési elveket.

A migráció egy konkrét megvalósításának kidolgozásához példaként a Condor [AA] ütemező rendszert választottam, melyet a University of Wisconsin (Madison, USA) Számítástudományi Intézetének munkatársai fejlesztettek ki. Az ütemező jelentősségét jól tükrözi, hogy napjainkra már több mint 800 klaszter, mintegy 100 ezer számítógépét felügyeli a világ számos intézetében.

A Condor rendszer működésének behatóbb tanulmányozásában segítségemre volt a wisconsin-i egyetem fejlesztőivel való szoros együttműködés tanulmányutak és projektek keretében. A Condor sajátosságainak megismerését követően dolgoztam ki az általam bemutatott folyamat- illetve alkalmazás migrációs eljárást.

III. Új tudományos eredmények

1 Új ellenőrzőpontozási módszer KlaszterGrid környezetben

Egy üzenetközvetítésen alapuló párhuzamos algoritmus, illetve alkalmazás ellenőrzőpontozása több módon is megvalósítható. Egy konkrét megvalósításnak a szoftverkörnyezet által támasztott igényeket minden esetben ki kell elégítenie. Az első téziscsoport célja, hogy azonosítsa egy általános KlaszterGrid környezet által az ellenőrzőpontozási és állapot-átviteli technikákkal szemben támasztott igényeket, majd kidolgozzon egy, a követelményeket kielégítő absztrakt módszert.

Az 1.1 tézis előkészítéseként meghatároztam a KlaszterGrid környezet főbb jellemzőit, illetve az állapot-átviteli mechanizmus elérni kívánt működési eseteit. A megfelelő követelmények meghatározása érdekében, azonosítottam a működést befolyásoló komponenseket, majd definiáltam a követelményeket 4+1 pontban. Az ASM formális módszer segítségével modelleztem a Klaszter komponenseit és azok főbb jellemzőit, majd erre alapulva formalizáltam az ellenőrzőpontozási technikákkal szemben felállított követelmény rendszert. Végül a követelményrendszer segítségével elvégeztem a vizsgált megoldások kiértékelését. Az eredmények alapján megállapítottam az 1.1 tézist.

> **1.1 tézis:** *Üzenetközvetítéssel kommunikáló párhuzamos algoritmusok esetén Klaszter környezetben meghatározható olyan formális feltételrendszer, mely transzparens ellenőrzőpontozást tesz lehetővé az előzőekben definiált működési esetekre vonatkozóan. Továbbá jelenleg nem érhető el a működési eseteket megvalósító ellenőrzőpontozási és állapot-átviteli módszer a meghatározott körülmények között.*
>
> A disszertáció kapcsolódó fejezete: 2.1
>
> Kapcsolódó publikációk: [3][4][17]

Az 1.1 tézisben felállított követelményrendszer definiálja a működési kritériumokat a transzparens ellenőrzőpontozást megcélzó megoldások számára. A

különböző ellenőrzőpontozási technikák által kialakítható tervezési térben a konkrét megoldás eléréséhez, meghatároztam a kívánt ellenőrzőpontozási módszer alapvető működésbeli, felépítésbeli sajátosságait. Ennek eredményeképpen egy 7 pontból álló új módszert dolgoztam ki, melyet az 1.1 tézisben bevezetett formális modellbe (CP_{ground}) illesztettem. Továbbá a módszer belső működési mechanizmusának leírására ASM szabályokat dolgoztam ki. Végezetül absztrakt események bevezetésével igazoltam az 1.2 tézist.

1.2 tézis: *A CP_{ground} ASM modell segítségével definiált új ellenőrzőpontozási módszer transzparens működést valósít meg, mind a szoftver környezet felé, mind a programozó számára, egyidejűleg.*

A disszertáció kapcsolódó fejezete: 2.2

Kapcsolódó publikációk: [3][4][5][17]

A CP_{ground} absztrakt modell segítségével meghatározott módszer lehetővé teszi egy üzenetközvetítésen alapuló párhuzamos algoritmus állapotterének konzisztens, transzparens módon történő lementését és visszaállítását.

2 Ellenőrzőpontozási módszer alkalmazása az üzenetküldő párhuzamos algoritmusok számára

Az első téziscsoportban bemutatott elméleti háttér – mely egy absztrakt módon meghatározott ellenőrzőpontozási módszert eredményezett – megfelelő alapot ad egy konkrét eszköz elkészítésére. Az elmélet gyakorlati hasznosítására az MTA SZTAKI által kifejlesztett P-GRADE Grafikus Párhuzamos Programozási környezet megfelelő választásnak bizonyult. A második téziscsoport célja alkalmazni az absztrakt modellt a – P-GRADE fejlesztőeszköz által is támogatott – PVM üzenetközvetítésen alapuló algoritmusok számára.

A 2.1 tézis előkészítése érdekében megterveztem és megvalósítottam az 1.2 tézisben kidolgozott absztrakt ellenőrzőpontozási módszert a P-GRADE fejlesztői környezetben készülő PVM alkalmazások számára. Tanulmányoztam a GRAPNEL alkalmazások felépítését, definiáltam ennek módosítását és megoldottam a kommunikációs primitívek oly módon történő átdolgozását, mely lehetővé teszi az ellenőrzőpontozási művelet tetszőleges időpillanatban történő megvalósítását. Emellett a P-GRADE-ben bemutatott megoldáshoz kidolgoztam a hozzá illeszkedő absztrakt ($CP_{grapnel}$) modellt, majd egy általam kidolgozott finomítási módszerrel bizonyítottam a CP_{ground} és $CP_{grapnel}$ közötti összefüggést. Az eredmények alapján megállapítottam a 2.1 tézist.

2.1 tézis: *Statikus folyamatmodellt követő párhuzamos GRAPNEL alkalmazásokon a P-GRADE rendszerben kialakított ellenőrzőpontozás transzparens, továbbá a GRAPNEL ellenőrzőpontozó esetében alkalmazott $CP_{grapnel}$ modell a CP_{ground} modellnek egy korrekt finomítása.*

A disszertáció kapcsolódó fejezete: 3.1

Kapcsolódó publikációk: [1][3][13][15][16]

A megoldás P-GRADE fejlesztő környezetben statikus folyamat modellen alapuló alkalmazások számára nyújt –a futtató környezet és a programozó számára egyaránt– transzparens ellenőrzőpontozási lehetőséget.

A 2.2 tézis előkészítéseként megterveztem és megvalósítottam az 1.2 tézisben kidolgozott absztrakt ellenőrzőpontozási módszert általános PVM alkalmazások számára a 2.1 tézis eredményeinek általánosításával. Tanulmányoztam a PVM alkalmazások jellegzetességeit, a PVM által nyújtott szolgáltatásokat. Definiáltam a PVM alkalmazás struktúráját és megoldottam a kommunikációs primitívek megszakíthatóságának problémáját. A bemutatott megoldáshoz kidolgoztam a hozzá illeszkedő absztrakt ASM modellt (CP_{tckpt}). Végezetül a TotalCheckpoint (TCKPT) által megvalósított CP_{tckpt} modellnek egy általam kidolgozott finomításával bizonyítottam a CP_{tckpt} és $CP_{grapnel}$ modell közti összefüggést. Az eredmények alapján meghatároztam a 2.2 tézist.

2.2 tézis: *Dinamikus folyamatmodellt követő PVM alkalmazásokhoz a TotalCheckpoint rendszer transzparens ellenőrzőpontozást valósít meg, továbbá a TCKPT esetében alkalmazott CP_{tckpt} modell a $CP_{grapnel}$ modellnek – és a tranzitivitás által a CP_{ground} modellnek is – egy korrekt finomítása.*

A disszertáció kapcsolódó fejezete: 3.2

Kapcsolódó publikációk: [4][5][8][9]

A 2.2 tézisben bemutatott megoldás natív PVM alkalmazások számára nyújt – a futtató környezet és a programozó számára egyaránt – transzparens ellenőrzőpontozási lehetőséget.

3 Állapottér átvitel az üzenetküldő párhuzamos algoritmusok számára

Az első téziscsoportban bemutatott elméleti háttérre, illetve a második téziscsoportban kidolgozott konkrét ellenőrzőpontozó módszerekre és eszközökre támaszkodva ennek a téziscsoportnak a célja igazolni, hogy KlaszterGrid környezetben transzparens állapottér átvitel (migráció) megvalósítható. Ennek egyik alternatívája az alkalmazás bizonyos folyamatainak migrációja a végrehajtó klaszter egyes erőforrásai között, másik alternatívája a teljes alkalmazás migrációja a klaszterek között. A kidolgozott módszerek bemutatására egy már elterjedt klaszter ütemezőt, a Condor rendszert választottam.

A 3.1 tézis a transzparens folyamat migráció témakörére fókuszál. Teszi mindezt oly módon, hogy a 2.1 tézisben kidolgozott transzparens ellenőrzőpontozást integráló GRAPNEL alkalmazás és a Klaszterek erőforrásainak ütemezését végző

Condor ütemező együttműködését vizsgálja meg. A tézisben ismertetem a Condor eszköz sajátosságait, meghatározom az alapvető működési feltételeket, a rendszer komponenseit, majd kidolgozom a migrációs mechanizmus lépéseit. A mechanizmus elméletét igazolom a $CP_{grapnel}$ modell-ből származtatott állapotátmenet diagram analízisével és a migráció lefolyásának modellben történő leképezésével. Az ismertetett eredmények segítségével igazoltam a 3.1 tézist.

3.1 tézis: *A P-GRADE eszköz által készített GRAPNEL alkalmazások migrációja megvalósítható az ütemező számára transzparens módon, továbbá az eredményezett megoldás illeszkedik a $CP_{grapnel}$ modell belső szabályhalmazára.*

A disszertáció kapcsolódó fejezete: 4.1

Kapcsolódó publikációk: [10][11][12][18]

Az ismertetett megoldás során a transzparencia abban mutatkozik meg, hogy a migrációs folyamat lejátszásához nem szükséges beavatkozás/módosítás a Condor, a PVM, az operációs rendszer komponensekbe, illetve az alkalmazás forráskódjába sem.

A 3.1 tézisben bemutatott folyamat migrációt egy – a GRAPNEL alkalmazásba épített – kitüntetett folyamat vezényli, melynek leállítása az alkalmazás leállását eredményezi. Viszont a KlaszterGrid egyes klaszterei között kapcsolat nincs, ezért e folyamat leállítására mégis szükség van a két klaszter közötti teljes alkalmazás migráció során. Erre a célra a tézisben egy alkalmazás migrációs eljárást mutatok be, majd a mechanizmus elméletét igazolom a $CP_{grapnel}$ modellbe történő leképezéssel. Az eredményekre alapozva megfogalmaztam a 3.2 tézist.

3.2 tézis: *A GRAPNEL alkalmazás megvalósítja a konzisztens állapottér átvitelt üzenetközvetítéssel kommunikáló párhuzamos algoritmusok számára a beépített szerver folyamat állapotának lementésével, ezáltal lehetővé téve a párhuzamos GRAPNEL alkalmazások független (egymás erőforrásait nem használó) klaszterek közötti migrációját is. Továbbá az eredményezett megoldás illeszkedik a $CP_{grapnel}$ modell belső szabályhalmazára.*

A disszertáció kapcsolódó fejezete: 4.2

Kapcsolódó publikációk: [1][10][13][14]

IV. Az eredmények alkalmazása

Az ellenőrzőpontozási technika napjainkban egyre nagyobb fontossággal bír, mivel ez elengedhetetlen egy hosszan futó alkalmazás hibatűrő végrehajtásához. Nagy mennyiségű erőforrást tartalmazó Grid rendszerben az erőforrás leállását okozó hiba előfordulásának valószínűsége nem elhanyagolható. Hosszan futó (hetek/hónapok) alkalmazások esetén pedig igen nagy a valószínűsége, hogy legalább egyszer erőforrás vagy hálózati hiba következtében leáll a végrehajtás [BB]. Egy ilyen hiba az

addig elvégzett futás eredményeinek megsemmisülését eredményezheti. A hasznos futási idő elvesztésének kiküszöbölése [J] önmagában hatalmas eredményt jelent a Grid számára, melynek komoly gazdasági előnye is van. A tézisekben bemutatott eredmények bármely ClusterGrid környezetben felhasználhatóak, mivel támogatják az infrastruktúra sajátosságaitól való függetlenséget.

A P-GRADE fejlesztő eszköz több európai egyetemen került bemutatásra, illetve lett oktatási célokra felhasználva. Emellett számos kapcsolódó kutatási téma zajlik napjainkban is, melyet ezen eszköz indukált párhuzamos programozás témakörben. A P-GRADE rendszert sikeresen alkalmazták a meteorológiában egy ultrarövidtávú időjárás előrejelző rendszer párhuzamosítására (Országos Meteorológiai Szolgálat) [CC], a mérnöki tudományok területén városi forgalom szimulációra (University of Westminster), valamint reakció-diffúzió rendszerek modellezésére a kémiában (ELTE) [DD]. Mivel a P-GRADE fejlesztői környezet integrált részét képezi az ellenőrzőpontozó és migrációs eszköz is, az itt kifejlesztett alkalmazások automatikusan kiegészülnek az eszköz által nyújtott képességekkel.

A tézisekben bemutatott eredményeket számos konferencia kiadványban és folyóiratban publikáltam, valamint több tudományos fórumon és kiállításon kerültek bemutatásra. A publikációk további kutatásokat inspiráltak, a bemutatott munka számos hazai és nemzetközi együttműködés alapját képezte. Hazaiak közül kiemelendő az országos KlaszterGrid [EE] infrastruktúrát (~30 intézet, ~1800 gép, ~400GFlops csúcsteljesítmény) üzemeltető NIIF intézettel való közös munka a SzuperKlaszter (IKTA) projektben, mely során a bemutatott TotalCheckpoint ellenőrzőpontozó rendszer került kifejlesztésre az infrastruktúra számára. Nemzetközi együttműködések egyik legjelentősebbike a CoreGRID EU kiválósági hálózat (Network of Excellence), ahol a tézisekben bemutatott eredményeket egy átfogó Grid Ellenőrzőpontozási Rendszer (GCA) [6][7][19][20][21] részévé kívánják tenni az együttműködő partnerek.

V. Publikációs lista

Folyóiratcikkek

[1] P. Kacsuk, G. Dózsa, **J. Kovács**, R. Lovas, N. Podhorszki, Z. Balaton and G. Gombás: "*P-GRADE: a Grid Programming Environment*", Journal of Grid Computing Vol. 1. No. 2, pp. 171-197. 2004.

[2] **Kovács, J.**, Kacsuk, P.: "*The DIWIDE Distributed Debugger*", Quality of Parallel and Distributed Programs and Systems, special issue of Journal of Parallel and Distributed Computing Practices, PDCP Vol.4, No. 4, Eds: P.Kacsuk, G.Kotsis, pp. 331-347, 2001

[3] **J. Kovacs**: "*Transparent Parallel Checkpointing and Migration in Clusters and ClusterGrids*", International Journal of Computational Science and Engineering, IJCSE, 2006, (to appear)

[4] **József Kovács**, Rafal Mikolajczak, Radoslaw Januszewski, Gracjan Jankowski: "*Application and Middleware Transparent Checkpointing with TCKPT on ClusterGrid*", Future Generation Computer Systems, selected papers of DAPSYS2006, (accepted)

Konferenciacikkek

[5] **J. Kovacs**, R. Mikolajczak, R. Januszewski, G. Jankowski: "*Application and middleware transparent checkpointing with TCKPT on Clustergrid*", Proceedings of 6th Austrian-Hungarian Workshop on Distributed And Parallel Systems, DAPSYS 2006, Innsbruck, Austria, September 21-23, 2006, pp. 179-189.

[6] G. Jankowski, **J. Kovacs**, R. Mikolajczak, R. Januszewski, N. Meyer: "*Towards Checkpointing Grid Architecture*", Parallel Processing and Applied Mathematics – Conference on Parallel Processing and Applied Mathematics, PPAM2005, Poznan, Poland, Lecture Notes in Computer Science, Vol. 3911/2006, pp. 659-666, Springer, 2006, ISBN 978-3-540-34141-3

[7] G. Jankowski, R. Januszewski, **J. Kovacs**, N. Meyer, R. Mikolajczak: "*Grid Checkpointing Architecture - a revised proposal*", Proc. of the 1st CoreGRID Integration Workshop, pp. 287-296, Pisa, 28-30, November, 2005

[8] **Kovács József**, Farkas Zoltán, Marosi Attila: "*Ellenőrzőpont támogatás PVM alkalmazások számára a magyar ClusterGriden*", Networkshop, Szeged, 2005

[9] **Jozsef Kovacs**: "*Making PVM applications checkpointable for the Grid*" Proc. of the Microcad 2005 Conference, Section N, Miskolc, 2005, pp. 223-228

[10] **József Kovács**: „*Process Migration in Clusters and Cluster Grids*", Distributed and Parallel Systems: Cluster and Grid Computing, Kluwer International Series in engineering and Computer Science, Vol. 777, Dapsys 2004, Budapest, Hungary, pp. 103-110.

[11] **József Kovács**, Péter Kacsuk: "*A migration framework for executing parallel programs in the Grid*", In: Grid Computing – Second European AcrossGrids Conference, AxGrids 2004, Nicosia, Cyprus, Lecture Notes in Computer Science, Vol. 3165, pp. 80-89, Springer-Verlag, 2004

[12] **József Kovács**, Péter Kacsuk: "*Improving fault-tolerant execution for parallel applications under Condo*r", microCAD International Scientific Conference, University of Miskolc, Miskolc, Hungary, March 18-19, 2004, pp. 251-256

[13] **József Kovács**, Péter Kacsuk: "*Párhuzamos programok vándorlása a Grid-en*", University of Miskolc, Doktoranduszok fóruma, Gépészmérnöki kar szekciókiadványa, 2003, pp. 158-164

[14] P. Kacsuk, R. Lovas, **J. Kovács**, G. Dózsa, N. Podhorszki: "*Metacomputing support by P-GRADE*", GGF8 Workshop on Grid Applications and Programming Tools, 2003

[15] **Jozsef Kovacs**, Peter Kacsuk: "*Server based migration of parallel applications*", 4th DAPSYS Conference, Linz, Austria, 29th September-2nd October 2002, pp: 30-37

[16] **József Kovács**: "*Párhuzamos programok checkpointolása és migrációja klasztereken*", Networkshop'2002, Eger, Eszterházy Károly Főiskola, 26th-28th March 2002

Konferencia előadások és poszterek

[17] **Jozsef Kovacs**: "*Formal analysis of existing checkpointing systems and introduction of a novel approach*", CSCS 2006, Szeged, Hungary, June 2006 (honored with Best Talk Award)

[18] **Jozsef Kovacs**: "*PVM & Condor checkpointing*", Condor Week 2004, April 14-16, 2004, University of Wisconsin, Madison

Technikai riportok

[19] G. Jankowski, R. Januszewski, R. Mikolajczak, **J. Kovacs**: "*Scalable multilevel checkpointing for distributed applications - on the integration possibility of TCKPT and psncLibCkpt*", CoreGRID Technical Report, TR-0019, March 2006

[20] G. Jankowski, R. Januszewski, R. Mikolajczak, **J. Kovacs**: "*Scalable multilevel checkpointing for distributed applications - on the possibility of integrating Total Checkpoint and AltixC/R*", CoreGRID Technical Report, TR-0035, May 2006

[21] G. Jankowski, R. Januszewski, R. Mikolajczak, **J. Kovacs**: "*Grid Checkpointing Architecture - a revised proposal*", CoreGRID Technical Report, TR-0036, May 2006

VI. REFERENCIÁK

[A] I. Foster, C. Kesselman, S. Tuecke, “The Anatomy of the Grid. Enabling Scalable Virtual Organizations”, International Journal of Supercomputer Applications, 15(3), 2001

[B] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003

[C] Condor homepage: http://www.cs.wisc.edu/condor

[D] V. S. Sunderam: ”PVM: A Framework for Parallel Distributed Computing”, Concurrency: Practice and Experience, 2, 4, pp.:315-339, December, 1990.

[E] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Mancheck and V. Sunderam. PVM: Parallel Virtual Machine a User’s Guide and Tutorial for Networked Parallel Computing, MIT Press, Cambridge, MA, 1994.

[F] William Gropp, Ewing Lusk, and Anthony Skjellum: ”Using MPI”, 2nd Edition, MIT Press, ISBN 0-262-57132-3

[G] Al Geist, Ewing Lusk, William Gropp, William Saphir, Steve Huss-Lederman, Tony Skjellum, Andrew Lumsdaine, and Marc Snir.: ”MPI-2: Extending the Message-Passing Interface”, In EuroPar96, February 1996

[H] Elnozahy, E. N., Alvisi, L., Wang, Y., and Johnson, D. B. 2002. A survey of rollback-recovery protocols in message-passing systems. ACM Comput. Surv. 34, 3 (Sep. 2002), 375-408. DOI= http://doi.acm.org/10.1145/568522.568525

[I] S.Kalaiselvi and V.Rajaraman, A Survey of Checkpointing Algorithms for Parallel and Distributed Computers, Sadhana, Vol.25, Part 5, October 2000, pp.489-510

[J] M. Treaster. A Survey of Fault-Tolerance and Fault-Recovery Techniques in Parallel Systems. ACM Computing Research Repository (CoRR), (cs.DC/ 0501002), January 2005. http://citeseer.ist.psu.edu/treaster05survey.html

[K] P-GRADE Parallel Grid Application Development and Execution Environment, http://www.lpds.sztaki.hu/pgrade

[L] D. Drótos, G. Dózsa, and P. Kacsuk, “GRAPNEL to C Translation in the GRADE Environment”, Parallel Program Development for Cluster Comp. Methodology, Tools and Integrated Environments, Nova Science Publishers, Inc. pp. 249-263, 2001

[M] George Stellner, ”Consistent Checkpoints of PVM Applications”, In Proc. 1st Euro. PVM Users Group Meeting, 1994

[N] J. Leon, A. L. Fisher, and P. Steenkiste, ”Fail-safe PVM: a portable package for distributed programming with transparent recovery”. CMU-CS-93-124. February, 1993

[O] Iskra, K. A., van der Linden, F., Hendrikse, Z. W., Overeinder, B. J., van Albada, G. D., and Sloot, P. M. 2000. The implementation of dynamite: an environment for migrating PVM tasks. SIGOPS Oper. Syst. Rev. 34, 3 (Jul. 2000), 40-55. DOI= http://doi.acm.org/10.1145/506117.506123

[P] J. Casas, D. Clark, R. Konuru, S. Otto, R. Prouty, and J. Walpole, ”MPVM: A Migration Transparent Version of PVM”, Technical Report CSE-95-002, 1, 1995

[Q] C.P.Tan, W.F. Wong, and C.K. Yuen, ”tmPVM - Task Migratable PVM”, In Proceedings of the 2nd Merged Symposium IPPS/SPDP, pp. 196-202, 1999.

[R] Pawel Czarnul: ”Programming, Tuning and Automatic Parallelization of Irregular Divide-and-Conquer Applications in DAMPVM/DAC” in International Journal of High Performance Computing Applications, 2003, Vol.17, No.1

[S] Dan Pei, Wang Dongsheng, Zhang Youhui, Shen Meiming, ”Quasi-asynchronous Migration: A Novel Migration Protocol for PVM Tasks.” ACM SIGOPS Operating Systems Review, 33(2): 5-15 (April 1999).

[T] E. Börger, “High level system design and analysis using abstract state machines”, ASM Workshop, Magdeburg, September 1998

[U] E. Börger, “Why use evolving algebras for hardware and software engineering?”, SOFSEM’95, LNCS, 1995

[V] Y.Gurevich, “Evolving algebras 1993: Lipari guide”, E.Börger, editor, Specification and Validation Methods, pages 9-36, Oxford University Press, 1995

[W] E. Börger and D. Rosenzweig, “The WAM – definition and compiler correctness”, Logic Programming: Formal Methods and Practical Applications, 1994

[X] E. Börger and I. Durdanovic, “Correctness of compiling occam to transputer code”, Computer Journal, 39(1):52-92, 1996

[Y] E. Börger and W. Schulte, “Programmer friendly modular definition of the semantics of java”, Formal Syntax and Semantics of Java, LNCS, Springer, 1998

[Z] Mike Barnett, Egon Boerger, Yuri Gurevich, Wolfram Schulte and Margus Veanes, “Using Abstract State Machines at Microsoft: A Case Study”, Proceedings of ASM'2000 in "Abstract State Machines: Theory and Applications" Eds. Y. Gurevich, P. Kutter, M. Odersky, and L. Thiele Springer Lecture Notes in Computer Science vol. 1912, pages 367--379, 2000

[AA] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003

[BB] Ralston A, Reily E D 1993 Encyclopedia of computer science 3rd edn (New York: IEEE Press)

[CC] R. Lovas, P. Kacsuk, A. Horvath, A. Horanyi: Application of P-GRADE Development Environment in Meteorology, In: Distributed and Parallel Systems, Cluster and Grid Computing, pp. 109-116, Kluwer Academic Publishers, 2002

[DD] R. Lovas, P. Kacsuk, I. Lagzi, T. Turányi: „Unified development solution for cluster and grid computing and its application in chemistry”, In: Computational Science and Its Applications – ICCSA 2004: International Conference, Assisi, Italy, Lecture Notes in Computer Science, Vol. 3044, pp. 226-235, Springer-Verlag, 2004

[EE] NIIF ClusterGrid Project, http://www.clustergrid.niif.hu