Universitatea POLITEHNICA din Bucureşti

Facultatea de Automatică şi Calculatoare

Catedra de Calculatoare

ARHITECTURI ŞI TEHNICI DE PARALELIZARE PENTRU SERVERE DE SPAŢII VIRTUALE 3D MMO

ARCHITECTURES AND PARALLELIZATION TECHNIQUES FOR 3D MMO SERVERS

Rezumat

Doctorand:
Ing. Victor Asavei

Conducător ştiinţific:
Prof.Dr.Ing. Florica Moldoveanu

Bucureşti 2011

CUPRINS

1 INTRODUCERE

Jocurile de tip Massive Multiplayer Online (MMO) precum şi alte tipuri de spaţii virtuale cum ar fi muzee virtuale, expoziţii, etc, atrag un număr din ce în ce mai mare de utilizatori. Tehnologia cât şi numărul de aplicaţii de acest fel se află într-o continuă dezvoltare şi foarte probabil spaţiile virtuale 3D vor reprezenta următoarea generaţie de interfeţe de comunicare pe Internet, înlocuind sau integrând pe cele actuale (browsere de web, instant messengers, portaluri de socializare etc).
Aplicaţiile de tip MMO se desfăşoară online având ca suport o lume virtuală persistentă la care sunt conectaţi şi în cadrul căreia interacţionează simultan sute şi chiar mii de utilizatori.
Din punct de vedere al proiectării, unul dintre obiectivele principale ale unei aplicaţii MMO 3D este acela de a crea un grad de imersiune cât mai ridicat pentru utilizatorii spaţiului virtual - prin calitatea simulării (grafica 3D moderna , sunete stereo şi spaţiale, simulări realiste ale fizicii, etc.), dinamica interacţiunilor (timp real) şi atractivitatea scenariului.
Principalul atu al unei aplicaţii de tip MMO este însă dat de numărul mare de utilizatori pe care îl poate suporta simultan. Aceasta este principala caracteristică prin care se diferenţiază de aplicaţiile single-user / multiplayer care simulează o lume virtuală 3D.

1.1 MOTIVAŢIE ŞI OBIECTIVE

Principala problema a arhitecturilor curente pentru servere de spaţii virtuale 3D MMO este cea a scalabilităţii acestora, atunci când trebuie sa facă faţă unui număr mare de utilizatori simultan, numărul de operatii ce trebuie sa fie executate ajungând să fie extrem de ridicat în momentele de încărcare maximă. Aceste probleme de scalabilitate apar în mare masura datorita necesitatii de a mentine consistenţa lumii virtuale.

Marea majoritate a spaţiilor virtuale 3D MMO folosesc în prezent o arhitectură de tipul Client-Server. Deşi acest tip de arhitectură pune la dispoziţie funcţionalităţile necesare unei aplicaţii MMO o face cu un cost mare. Pentru a putea simula spaţii

virtuale complexe, în cadrul arhitecturii Client-Server se folosesc un număr ridicat de echipamente pentru partea de server şi în ciuda numărului lor mare, clusterele de servere sunt limitate din punct de vedere computaţional iar traficul de reţea introdus de comunicarea cu clienţii nu este nici el de neglijat.
Costurile mari introduse de arhitectura Client-Server limitează practic scalabilitatea acesteia, fiind necesară împărţirea spaţiului virtual în mai multe versiuni distincte ale aceleiaşi lumi virtuale. Deşi aceasta soluţie permite un anumit grad de scalabilitate al aplicaţiilor MMO, ea limiteaza gradul de realism împiedicând interacţiunea unui număr mare de utilizatori.
De asemenea, pentru a obține un nivel de performanţă în prezenţa unui număr mare de utilizatori, producătorii spaţiilor virtuale MMO trebuie sa facă faţă unui cost financiar semnificativ pentru a menţine infrastructura sistemului.

Ţinând cont de aceste limitări, obiectivele principale ale lucrării de faţă sunt următoarele :

- Analiza soluţiilor tehnologice / arhitecturilor actuale folosite pentru redarea spaţiilor virtuale 3D atât la nivelul clientului cât şi la cel al serverului
- Identificarea operaţiilor ce pot fi optimizate, atât la nivelul clientului cât şi la cel al serverului, şi implementarea acestor optimizări folosind tehnici de paralelizare GPGPU
- Propunerea unei soluţii arhitecturale originale pentru a elimina sau reduce o parte din problemele curente de scalabilitate prin :
    - Introducerea în arhitectura serverelor a capabilităţilor de procesare a sarcinilor de calcul folosind tehnici GPGPU
    - Folosirea unui mecanism de alocare dinamică a resurselor care va ţine cont de o serie de factori asociaţi utilizatorilor spaţiului virtual şi operaţiilor executate de aceştia
- Implementarea şi testarea unui prototip funcţional al soluţiei arhitecturale propuse

1.2 PREZENTAREA PE SCURT A CONŢINUTULUI FIECĂRUI CAPITOL

1. **SPAŢII VIRTUALE 3D MMO – CONCEPTE ŞI TEHNOLOGII**

Acest capitol prezintă conceptele specifice spaţiilor virtuale 3D MMO. De asemenea sunt trecute în revistă pe scurt şi tehnologiile folosite pentru redarea 3D a spaţiului virtual la client cât şi cele folosite de servere pentru a realiza comunicarea în reţea şi a asigura accesul controlat al utilizatorilor la spaţiul virtual.

Noţiunile descrise în acest capitol au fost prezentate şi într-o carte la care autorul lucrării de faţă a fost co-autor *([MMA09a]*).

2. **ANALIZA SOLUŢIILOR ARHITECTURALE ACTUALE PENTRU SERVERE DE SPAŢII VIRTUALE 3D MMO**

Acest capitol investighează soluţiile arhitecturale utilizate în prezent în implementarea de spaţii virtuale 3D MMO. Este prezentată în detaliu funcţionarea arhitecturii Client-Server care este cea mai folosită soluţie utilizată în acest moment de aplicaţiile MMO. De asemenea este trecut în revistă şi modul de funcţionare al arhitecturii Peer-To-Peer.

În acest capitol sunt identificate principalele avantaje dar şi lipsuri pentru fiecare din arhitecturile actuale, aceste neajunsuri apărând mai ales din cauza problemelor de scalabilitate ce sunt generate de numărul ridicat de utilizatori ce accesează în acelaşi timp spaţiul virtual.

3. **TEHNICI INOVATIVE DE PARALELISM PENTRU SPAŢII VIRTUALE 3D MMO BAZATE PE GPGPU**

Acest capitol prezintă la începutul său o descriere a tehnologiei GPGPU (General Purpose on Graphical Processing Units), de ce a fost aleasă această metodă pentru optimizarea operaţiilor efectuate în spaţiile virtuale 3D MMO şi cum se poate implementa paralelizarea operaţiilor folosind arhitectura CUDA.

În continuarea capitolului sunt identificate operaţiile ce pot fi optimizate la nivelul serverului cât şi al clientului, propunându-se pentru acestea soluţii originale de implementare ce folosesc paralelismul atât la nivel single-GPGPU cât şi la nivel multi-GPGPU.

Pentru fiecare dintre implementările realizate se prezintă rezultatele obţinute cât şi o analiză a acestora.

Soluţiile şi rezultatele obţinute au fost publicate într-o serie de lucrări ştiinţifice (*[AMM09a] , [AMM09c], [AMM10a] , [AMM11]*).

4. ARHITECTURĂ ORIGINALĂ PENTRU SERVERE 3D MMO UTILIZÂND PARALELISMUL GPGPU

În acest capitol se propun soluţii la nivel arhitectural pentru a reduce problemele de scalabilitate cu care se confruntă arhitecturile tradiţionale de servere de spaţii virtuale 3D MMO. Abordările propuse încearcă obţinerea unei scalabilităţi crescute prin introducerea de capabilităţi de procesare single şi multi GPGPU la nivelul serverelor de spaţii virtuale 3D MMO cât şi a unui mecanism de alocare dinamică a resurselor necesare strâns legat de procesarea GPGPU.

Soluţiile propuse şi arhitectura ce le foloseşte au fost prezentate într-o serie de lucrări ştiinţifice ([AMB10],[MMA09b],[MMA09c]).

5. TESTAREA SCALABILITĂȚII SOLUȚIEI

În acest capitol se prezintă rezultatele unor teste ce vizează scalabilitatea soluţiei arhitecturale prezentate în capitolul anterior. În acest scop a fost realizat un prototip funcţional al arhitecturii şi au fost implementate atât aplicaţia server cât şi aplicaţia client necesară testării.

Capitolul se încheie prin prezentarea şi analiza rezultatelor obţinute, care sunt foarte încurajatoare, reprezentând astfel o primă evaluare pozitivă a soluţiilor propuse.

6. CONCLUZII, CONTRIBUȚII ORIGINALE ŞI EVOLUȚII VIITOARE

În partea finală a lucrării se realizează o sinteză a activităţilor de cercetare şi a rezultatelor obţinute.

De asemenea, se evidenţiază contribuţiile originale ce au fost aduse de către autorul tezei.

2 SPAŢII VIRTUALE 3D MMO – CONCEPTE ŞI TEHNOLOGII

Majoritatea spaţiilor virtuale de tip MMO care există în prezent, sunt jocuri virtuale comerciale de tipul MMOG (massively multiplayer online game).

Aşa cum spune şi numele, aplicaţiile MMOG permit accesul simultan al unui număr mare de jucători, în prezent acest număr variind de la sute până la zeci de mii în cadrul aceleiaşi lumi virtuale. Conexiunea acestora se realizează prin intermediul Internetului, aceste spaţii virtuale fiind practic accesibile oricui în mod gratuit sau cu costuri foarte reduse.

2.1 CONCEPTE GENERALE

Următoarele concepte generale sunt cele mai des utilizate în cadrul jocurilor MMOG :

1) **Avatar**

Avatarul este reprezentarea utilizatorului în lumea virtuală, cel mai adesea sub forma unui personaj 3D sau 2D animat.

Avatarul este vizibil atât utilizatorului caruia ii aparţine cât şi celorlalţi utilizatori.

Utilizatorul are posibilitatea de a-şi controla propriul avatar într-un mod mai mult sau mai puţin avansat. De asemenea, există posibilitatea de a particulariza reprezentarea avatarului, prin setarea unor trăsături fizice sau modificarea hainelor avatarului, alegerea vocii, etc.

2) **Personaj**

Foarte apropiat de conceptul de avatar este cel de ***personaj*** sau ***caracter***. Acesta reprezintă utilizatorul din punct de vedere al logicii lumii virtuale respective.

În general, mare parte din timpul de joc este dedicată îmbunătăţirii nivelului, abilităţilor şi posesiunilor personajului.

3) Monştrii

Aceste lumi virtuale nu sunt populate doar de avatarele jucătorilor ci şi de alte categorii de entităţi, cum ar fi monştrii sau NPC (non player characters).

Termenul consacrat „mobs" se referă la entităţi cu care utilizatorul este oarecum obligat de logica jocului să le confrunte pentru a îndeplini anumite obiective sau obține anumite resurse.

Acţiunile acestor entităţi sunt coordonate de către sistem, în general cu ajutorul unor elemente decizionale foarte simple.

4) NPC (Non-player characters)

NPC sunt entităţi ce populează lumea virtuală şi nu se află sub controlul utilizatorului.

NPC nu sunt în general inamici ce trebuie distruși ci au un rol mai variat, spre exemplu ei putând fi folosiţi pentru implementarea anumitor dialoguri cu variante predefinite sau pentru a oferi anumite produse sau servicii în cadrul jocului.

5) Posesiuni virtuale

Fiecare avatar poate avea anumite posesiuni virtuale, asupra cărora în general are drepturi absolute sau limitate în cadrul jocului respectiv. El poate utiliza, modifica, vinde sau distruge posesiunile sale. Conceptul de posesiune virtuală a dus şi la crearea unor sisteme de schimb/tranzacționare a acestora, majoritatea jocurilor MMOG incluzând piețe virtuale sau sisteme de licitații („auction-house"), etc.

6) Quest-uri

Un *quest* este o suită de acţiuni pe care utilizatorul trebuie să le îndeplinească pentru a obține o recompensă în cadrul lumii virtuale.

Acţiunile pot fi eliminarea unui număr de monştri de un anumit tip, eliminarea unui monstru foarte puternic, ieşirea dintr-un labirint, cautarea şi găsirea unui obiect special, asamblarea unui obiect din componente, etc.

7) Grupuri de utilizatori

Utilizatorii se pot organiza în diverse tipuri de grupuri, pentru a colabora în cadrul jocului. Grupurile de jucători pot avea caracter : temporar (echipe/teams/party) sau persistent (ghilde)

2.2 TEHNOLOGII CLIENT-SIDE

Aşa cum s-a precizat şi anterior, un aspect foarte important, atunci când vorbim de simulările 3D, este acela al imersiunii utilizatorului în cadrul lumii virtuale. Pentru a realiza acest lucru, trebuie să existe pe partea de client o redare grafică care foloseşte tehnologii 3D moderne pentru a avea :

- o reprezentare foarte detaliată a lumii virtuale,
- efecte vizuale spectaculoase cum ar fi : fum, explozii, umbre dinamice, texturi detaliate, etc

OpenGL şi Direct3D, principalele biblioteci dedicate dezvoltării de aplicaţii grafice 3D, permit crearea şi redarea de spaţii 3D în care un observator virtual poate naviga folosind mouse-ul sau tastatura. Scena este descrisa prin modelele 3D ale obiectelor, pozitionarea acestora şi a observatorului în scena, proprietăţile de material ale suprafeţelor obiectelor, pozitionarea şi proprietăţile surselor de lumina, texturile care trebuie aplicate pe suprafeţele obiectelor, etc.

2.3 TEHNOLOGII SERVER-SIDE

În proiectarea protocolului de comunicaţie dintre server şi utilizatori, într-o aplicaţie spaţiu virtual care se execută în cadrul unei reţele pe mai multe calculatoare, se poate opta pentru abordarea “peer-to-peer” sau abordarea “client/server”.

2.3.1 ABORDAREA PEER-TO-PEER

În abordarea peer-to-peer, două sau mai multe calculatoare comunică între ele, fiecare rulând o copie a aplicaţiei şi schimbând informaţii despre starea aplicaţiei.

Aceasta abordare, deşi conduce la un trafic redus pe reţea, implică alte probleme cum ar fi, de exemplu, tratarea traficului pierdut. Aceasta este mult mai importantă în momentul în care ruleaza mai mult de o copie a aplicaţiei, deoarece nu se mai ştie cine are datele corecte despre starea aplicaţiei şi cine nu. Diferenţele între stările aplicaţiei la diferiţi utilizatori pot crea probleme, deoarece toate copiile aplicaţiei trebuie să se sincronizeze între ele.

2.3.2 ABORDAREA CLIENT/SERVER

În abordarea client/server, în sistem exista doi actori :

- maşina server
- clienţii

Maşina server este cea care rulează aplicaţia şi transmite către toți clienţii starea curenta a aplicației, pe baza căreia fiecare client isi actualizează imaginea spaţiului virtual. Clienţii afișează scena spaţiului virtual așa cum le-a fost transmisă de către server şi redau sunetele pe care server-ul "le spune" să le redea.

Efectiv, clienţii se comporta ca nişte terminale care transmit intrările către server şi îl lasă pe acesta să se ocupe de aproape tot. În practică, acest mod de împărţire a responsabilităţilor nu este un lucru chiar așa de rau cum pare la prima vedere, ocazional server-ul putând sa ceară unui client să indeplinească şi alte funcționalități ce ar putea sa cadă în seama sa.

Pot exista foarte multe variații în abordarea client/server însă ideile de bază sunt cele prezentate în continuare.

3 SOLUŢII ARHITECTURALE ACTUALE PENTRU SERVERE DE SPAŢII VIRTUALE 3D MMO

3.1 FUNCŢIONALITĂŢI DE BAZĂ

O arhitectură a unei aplicaţii MMO trebuie să ofere următoarele funcţionalităţi de bază:

- Autentificare utilizatori
- Gestionarea drepturilor de acces ale utilizatorilor în lumea virtuală
- Consistenţa lumii virtuale
- Gestiunea ordinii evenimentelor
- Propagarea evenimentelor către entităţile spaţiului virtual
- Stocarea securizată a caracterelor şi posesiunilor acestora
- Planificarea operaţiilor computaţionale
- Efectuarea calculelor de actualizare a stării lumii virtuale
- Timpi de raspuns scăzuţi
- Securitatea lumii virtuale pentru a preveni eventualele încercări de a trișa

3.2 ARHITECTURI TRADIȚIONALE CLIENT-SERVER

3.2.1 DESCRIERE GENERALĂ

Marea majoritate a aplicaţiilor de tip MMO folosesc o arhitectură de tipul Client-Server. Clienţii accesează spaţiul virtual prin intermediul unui *server de conectare* care apoi ii direcționează către un *shard server*.

Shard-urile reprezintă versiuni independente ale aceleiaşi lumi virtuale şi se folosesc pentru a îmbunătăţi scalabilitatea aplicaţiei. *Shard-urile* nu sunt sincronizate şi utilizatorii de pe un s*hard* nu pot interacţiona cu utilizatorii de pe alt *shard*.
Deşi această soluţie permite un anumit grad de scalabilitate al aplicaţiilor MMO, ea limitează gradul de realism împiedicând interacţiunea unui număr mare de utilizatori.
Unele aplicaţii MMO, cum ar fi, de exemplu, *Eve Online* încearcă găsirea de noi soluţii, care să nu folosească *shard-uri* şi să permită interacţiunea **tuturor** utilizatorilor lumii virtuale. Aceste încercări sunt inca în faza incipientă şi momentan depind foarte mult de particularităţile lumii virtuale.

Arhitectura client-server este în general preferată deoarece un sistem centralizat oferă în mod natural următoarele avantaje: consistenţa lumii virtuale, securitate ridicată, evitarea problemelor de sincronizare, implementare mai simplă

3.2.2 LIMITĂRI ŞI INCONVENIENTE

Deşi arhitecturile de tip Client-Server pun la dispoziţie funcţionalităţile necesare unei aplicaţii MMO, o fac însă cu un cost ridicat. Din nefericire, natura puternic centralizata a arhitecturii Client-Server introduce o sugrumare din punct de vedere al performanţei. În ciuda numărului lor, care poate fi foarte mare, clusterele de servere sunt limitate din punct de vedere computaţional şi traficul de reţea este concentrat practic în echipamentele din data center. Costurile mari introduse de arhitectura Client-Server limitează practic scalabilitatea acesteia, fiind necesara împărţirea spaţiului virtual în mai multe instanţe / *sharduri*. De asemenea, pentru a obţine un nivel de performanţă care să facă faţă unui număr mare de utilizatori, producătorii spaţiilor virtuale MMO se confruntă cu un cost financiar semnificativ pentru a menţine infrastructura sistemului funcţională.

3.3 ARHITECTURI P2P

3.3.1 DESCRIERE GENERALĂ

Aplicaţiile MMO la scară redusă, de cele mai multe ori, folosesc abordări P2P. Acestea, în mod uzual, partajează între participanţi încărcarea simulării unui mediu

virtual. Astfel, abordarea P2P permite utilizatorilor noi să ofere sistemului resurse suplimentare astfel încât acesta sa facă faţă încărcării suplimentare pe care aceştia o introduc, sistemul reușind astfel să scaleze corespunzător.
Un alt avantaj al acestei abordări este că dacă una dintre resurse se defectează, ceilalţi participanți pot prelua atribuțiile acesteia oferind astfel sistemului robustețe.
Traficul de reţea se desfăşoară în funcţie de necesităţile participanților permițând partajarea încărcării între utilizatorii implicaţi în mod direct.
Toate aceste caracteristici oferă de asemenea reducerea costurilor pentru proprietarul spaţiului virtual care practic reduce cheltuielile cu mententanţa spaţiului virtual.

3.3.2 LIMITĂRI ŞI INCONVENIENTE

Deşi arhitectura Peer-to-Peer atunci când este folosită pentru implementarea serverelor de spaţii virtuale 3D MMO oferă în mod natural avantajele descrise mai sus, ridică în acelaşi timp şi o serie de probleme care trebuie rezolvate :

- riscul de securitate care apare odată cu oferirea puterii de decizie aplicației ce rulează la utilizator, acesta putând modifica aplicaţia pentru a obține avantaje care nu îi sunt cuvenite;
- menţinerea unei lumi virtuale persistente care să aibă aceleaşi caracteristici pentru toți utilizatorii care fac parte din ea;
- ordonarea evenimentelor şi propagarea acestora către toți utilizatorii.

Ţinând astfel cont de lipsurile şi inconvenientele pe care arhitecturile actuale le prezintă , este necesară explorarea de noi soluţii privind arhitecturile pentru servere de spaţii virtuale 3D MMO şi de asemenea găsirea unor modalităţi de optimizare a operațiilor efectuate în cadrul simulării unui lumi virtuale pentru a elimina, sau măcar reduce, o parte din problemele curente cu care se confruntă aplicaţiile MMO.

4 TEHNICI INOVATIVE DE PARALELISM PENTRU SPAŢII VIRTUALE 3D MMO BAZATE PE GPGPU

GPGPU (General Purpose on Graphical Processing Units) reprezintă o metodă prin care unitatea de procesare grafică poate fi utilizată pentru a executa calcule/programe de uz general.

4.1 DE CE GPGPU

Unitatea de Prelucrare Grafică (GPU) modernă a devenit un hardware foarte versatil în domeniul arhitecturilor de calcul paralele multi-core. Aceste arhitecturi, care includ GPU şi CPU multi-core de la AMD şi Intel, procesoarele CELL , procesoarele SUN UltraSparc, se diferențiază faţă de arhitectura clasică CPU prin următorul fapt : favorizarea operațiilor ce se pot executa în paralel asupra unei cantităţi mari de date faţă de execuția operațiilor *single-task* de latenţă scăzută.
Deşi există diferenţe de implementare între diverşii producători, toate GPU-rile moderne încearcă să menţină o eficienţă ridicată prin folosirea de arhitecturi multi-core, care folosesc atât multithreading hardware cât şi procesare SIMD – Single Instruction Multiple Data. Aceste tehnici nu sunt unice pentru GPU însă în comparație cu CPU, GPU duc la extrem aceste arhitecturi.
De exemplu, GPU NVIDIA GeForce GTX590 are 1024 de procesoare scalare care operează la 1,2 GHz. Acestea sunt organizate în grupuri de 32 şi au un *peak-rate* (vârf de performanţă) de 2,5TFLOPS. În comparaţie, un procesor high-end de la Intel , XEON Westmere care operează la o frecvenţă de 3.3GHz conține 6 core-uri şi are un *peak-rate* de 146GFLOPS.

4.1.1 CUDA

CUDA (Compute Unified Device Architecture) este o arhitectură hardware şi software pentru realizarea şi gestionarea calculelor pe GPU, fără a fi nevoie de maparea la un API specializat pentru grafică.

În programarea realizată cu ajutorul CUDA, GPU este văzut ca un co-procesor la CPU principal, numit gazdă. Cu alte cuvinte, porțiuni care sunt intensiv computaţionale şi prezintă paralelism de date, din aplicaţia ce rulează pe gazdă, sunt încărcate pe dispozitiv (GPU).
Această porțiune din aplicație este organizată sub forma unei funcţii care este executată simultan de un număr mare de fire de execuție. Funcția destinată rulării pe GPU este compilată în instrucțiuni specifice dispozitivului GPU, rezultând un program numit *kernel*.

4.2 UTILIZAREA GPGPU PENTRU REDAREA 3D CLIENT-SIDE

Unul dintre obiectivele principale ale unei aplicaţii MMO 3D pe partea de client, este acela de a crea un grad de imersiune cât mai ridicat pentru utilizatorii spaţiului virtual. RayTracing reprezintă o metodă de redare, care utilizată în cadrul simulărilor 3D, poate să ofere un grad de realism net superior comparativ cu tehnologiile 3D clasice care utilizează ca metodă de redare banda grafică. În continuare se va prezenta, o soluţie pentru a executa algoritmul RayTracing folosind procesare GPGPU şi multi-GPGPU precum şi evaluarea rezultatelor obţinute.

4.2.1 ADAPTARE ALGORITM RAYTRACING PENTRU GPGPU

Algoritmul pentru împărțirea calculelor RayTracing pe firele de execuţie CUDA este următorul :

- Datele iniţiale ale scenei sunt făcute disponibile în memoria globală a GPU;
- Setul de raze este determinat de o matrice de pixeli ce are dimensiunea ecranului
- Matricea ecran va fi la rândul ei împărțită în sub-matrici ce au dimensiuni fixe (8,16 sau 32);

- Fiecare sub-matrice va fi procesată de un bloc de fire de execuţie unde fiecare fir de execuţie va fi responsabil de calculele executate pentru un element al sub-matricei (un pixel şi raza care trebuie sa treacă prin el);
- Pentru implementarea multi-GPGPU, imaginea se va împărţi la numărul de GPU, fiecare GPU având asignată o porţiune din imagine (de exemplu, dacă se folosesc două GPU, GPU0 va procesa jumătatea superioară a ecranului şi GPU1 va procesa jumătatea inferioară a ecranului)

4.2.2 EVALUAREA REZULTATELOR

Pentru implementarea pe CPU (Single Core – 1 fir de execuţie şi Quad Core – 4 fire de execuţie) a fost utilizat un CPU Intel Core2Quad Q6600 şi algoritmul RayTracing a fost rulat folosind 1 fir de execuţie, respectiv 4 fire de execuţie.

Pentru implementările GPGPU (single şi multi) au fost folosite 2 GPU-uri NVidia 8800 Ultra montate în acelaşi calculator şi toolkit-ul CUDA 2.2.

Au fost testate toate cele 3 implementări şi rezultatele obţinute sunt cele prezentate în continuare.

	FPS Min	FPS Max	FPS Mediu
CPU Single Core 1 fir de execuţie	1	3	2.083
CPU Quad Core 4 fire de execuţie	5	7	5.983
NVidia 8800Ultra 128 procesoare folosind GPGPU	8	10	8.967
2xNVidia 8800Ultra 256 procesoare folosind GPGPU	16	18	16.980

Tabel 4-1: Rezultate implementare RayTracing folosind GPGPU

Rezultatele obţinute demonstrează faptul că algoritmul RayTracing scalează foarte bine atunci când este executat paralel şi ca se obţine un spor de performanţă prin implementarea GPGPU şi Multi-GPGPU.

4.3 UTILIZAREA GPGPU ÎN CADRUL SERVERELOR MMO

4.3.1 FIZICA SPAŢIULUI VIRTUAL

Pentru a asigura consistenţa şi corectitudinea funcţionării spaţiului virtual, aplicaţiile MMO trebuie să execute, la nivelul serverului, un număr foarte mare de operaţii pentru a simula o fizică realistă a spaţiului virtual (detecţia coliziunilor, forţe de frecare, forţa de gravitaţie, etc).
Acestea reprezintă gama de operaţii care sunt cele mai frecvent efectuate de către serverele spaţiului virtual şi astfel sunt principalele consumatoare de timp de calcul.

4.3.2 IMPLEMENTAREA SIMULĂRILOR FIZICII FOLOSIND GPGPU

În practică, se utilizează pentru detecţia coliziunilor metode ce iau în consideraţie :

- fie poligoanele din care sunt alcătuite obiectele 3D
- fie o aproximare a obiectelor 3D folosind volume încadratoare

În continuare se propune o modalitate de implementare a acestor tipuri de operaţii folosind tehnici GPGPU.

4.3.2.1 Algoritmi

Au fost implementate următoarele calcule de fizică atât pe GPU cât şi pe CPU (pentru comparaţie) : detecţia coliziunilor între obiecte, procesarea forţelor de ciocnire în urma coliziunilor, calcule pentru forţa de gravitaţie. De asemenea, au fost folosite următoarele tipuri de obiecte asupra cărora au fost efectuate calculele de coliziune : Cuburi, Sfere, Cilindri. Aceste tipuri de obiecte reprezintă aproximările cele mai folosite în practică ca volume încadratoare a obiectelor 3D complexe pentru efectuarea calculelor de coliziune.

Algoritmul pentru lansarea calculelor de coliziune pe firele de execuţie CUDA este următorul :

- Datele iniţiale ale scenei sunt făcute disponibile în memoria globală a GPU-ului

- Setul de sarcini de coliziune este determinat de un vector unidimensional de dimensiune N*N (unde N este numărul de obiecte din scenă) deoarece se simulează coliziuni N-la-N între obiecte
- Acest vector se împarte în sub-vectori de dimensiune fixă ce au fiecare dimensiunea 32
- Fiecare sub-vector va fi procesat de un bloc de fire de execuţie unde fiecare fir de execuţie va fi responsabil de calculele executate pentru un element al sub-vectorului (obiectul identificat de firul de execuţie va fi testat pentru coliziuni cu toate celelalte obiecte ale scenei)
- Pentru implementarea *multi-GPGPU*, vectorul va fi împărţit la numărul de GPU-uri obţinându-se astfel pentru fiecare GPU o porţiune de care acesta este responsabil (de exemplu dacă se folosesc două GPU-uri, GPU0 va procesa prima jumătate a vectorului şi GPU1 va procesa cea de-a doua jumătate a vectorului)

4.3.3 REZULTATE

Rezultatele obţinute sunt cele din tabelele de mai jos.

	850 de obiecte		1750 de obiecte		4000 de obiecte	
	FPS	Timp calcule fizică / iteraţie	FPS	Timp calcule fizică / iteraţie	FPS	Timp calcule fizică / iteraţie
1 CPU Quad Core Q6600	130	16ms	24.7	67ms	2.9	332ms
Single GPGPU 1 core GPU GTX590	202	2ms	98	3.8ms	44.6	7ms
Multi GPGPU 4 cores GPU GTX590	200	2.3ms	96	4ms	44	7.2ms

Tabel 4-2: Rezultate implementare calcule fizică – Partea I

	8000 de obiecte		16000 de obiecte		32000 de obiecte	
	FPS	Timp calcule fizică / iteraţie	FPS	Timp calcule fizică / iteraţie	FPS	Timp calcule fizică / iteraţie
1 CPU Quad Core Q6600	0.7	1314ms	N/A	N/A	N/A	N/A
Single GPGPU 1 core GPU GTX590	21.1	18ms	8	79ms	2.4	310ms
Multi GPGPU 4 cores GPU GTX590	22.4	15ms	11.6	30ms	5.4	80ms

Tabel 4-3: Rezultate implementare calcule fizică – Partea a II-a

Rezultatele obţinute au scos în evidenţă următoarele aspecte majore :

- implementarea folosind CPU nu mai face faţă în timp real pentru un număr relativ mediu de obiecte existente în scenă; se poate deduce astfel că procesarea pe CPU a calculelor de fizică nu este scalabilă cu numărul de obiecte din scenă;
- implementarea ce foloseşte *single GPGPU* obţine sporuri de performanţă de ordinul zecilor faţă de CPU;
- implementarea ce foloseşte *multi GPGPU* obţine sporuri de performanţă de ordinul sutelor faţă de CPU;
- diferenţele de spor de performanţă între implementarea *single GPU* şi implementarea *multi GPU* sunt mici pentru un număr relativ mic de obiecte din scenă, datorită overhead-ului introdus de operaţiile de sincronizare şi de transfer de memorie (gazdă-dispozitiv şi dispozitiv-gazdă) ce trebuie efectuate pentru a gestiona mai multe GPU în sistem; adevăratul spor de performanţă se vede când numărul de obiecte din scenă creşte semnificativ, sporul de performanţă crescând simţitor.

5 ARHITECTURĂ ORIGINALĂ PENTRU SERVERE 3D MMO UTILIZÂND PARALELISMUL GPGPU

Una dintre principalele probleme cu care se confruntă dezvoltatorii de spaţii virtuale 3D MMO este cea a încărcării foarte mare la care trebuie sa facă faţă lumea virtuală. Aceasta încărcare este dată atât de numărul de sarcini, care este direct proporţional cu numărul de utilizatori on-line, cât şi de complexitatea acestor sarcini care depinde de particularităţile lumii virtuale simulate.

5.1 NECESITATEA MODIFICĂRII ARHITECTURII CLASICE

În mod tradiţional, marea majoritate a aplicaţiilor MMO sunt implementate folosind o arhitectură de tip client-server.

Deşi această arhitectură, care este foarte răspândită, este potrivită pentru multe tipuri de aplicaţii distribuite, introduce şi o serie de dezavantaje cele mai importante fiind : **Scalabilitatea**, **Redundanţa, Fiabilitatea şi Costul.**
În general, dezvoltarea unei aplicaţii MMO durează între 2-3 ani şi costul de dezvoltare pentru a lansa o aplicaţie MMO de calitate medie este unul ridicat. Un alt aspect de luat în consideraţie este faptul ca aplicaţiile MMO trebuie sa simuleze lumi virtuale de întindere foarte mare şi acest lucru face necesară existenţa unui număr ridicat de servere, uneori de ordinul sutelor, pentru a găzdui în mod complet mediul virtual. Se poate ajunge astfel ca dupa lansarea unei aplicaţii MMO, costurile de întreţinere să ajunga chiar şi până la 80% din totalul încasărilor.

5.2 PREZENTAREA MODIFICĂRILOR

Modificările propuse în continuare pentru a fi aplicate în cadrul arhitecturilor serverelor spaţiilor virtuale încearcă să adreseze această problemă din două direcţii:

- introducerea în arhitectura serverelor a capabilităţilor de procesare a sarcinilor de calcul folosind tehnici GPGPU;

- propunerea unui mecanism original de alocare a resurselor necesare execuţiei calculelor ţinându-se cont de o serie de factori ce pot influenţa alocarea care sunt centraţi in jurul utilizatorilor spaţiului virtual

5.2.1 OPTIMIZARE CE ŢINE CONT DE POZIŢIONAREA SPAŢIALĂ ACTUALĂ A UTILIZATORILOR

O primă optimizare este dată de faptul că la alocarea resurselor se va ţine cont de poziţionarea spaţială a utilizatorilor din cadrul spaţiului virtual.

Astfel, dacă într-o zona geografică se află la un moment dat un anumit număr de utilizatori, alocarea resurselor pentru acea zonă poate fi ajustată dinamic (crescută sau scăzută) în funcţie de numărul acestora.

În acest fel, nu se vor aloca resurse în mod inutil pentru zone unde nu există un număr mare de calcule de executat (din cauza lipsei sau a numărului mic de utilizatori ce pot genera operaţii de procesat), evitându-se astfel fenomenul de „înfometare" iar resursele disponibile vor putea fi utilizate în alte zone unde este nevoie de resurse suplimentare.

5.2.2 OPTIMIZARE CE ŢINE CONT DE COSTUL OPERAŢIILOR EFECTUATE DE FIECARE UTILIZATOR

O altă optimizare propusă ţine cont de tipul de operaţii efectuate de un utilizator şi de frecvenţa cu care acesta generează aceste operaţii.

Prin execuţia de operaţii de *profiling* în timp asupra acţiunilor unui utilizator din spaţiul virtual, se poate astfel determina un cost de „greutate a operaţiilor efectuate" asociat acestuia şi în funcţie de acest cost se poate schimba categoria din care face parte utilizatorul (utilizator care generează operaţii care nu sunt intensiv computaţionale, utilizator care generează operaţii care sunt intensiv computaţionale, etc).

În funcţie de apartenenţa la diversele categorii existente se poate face o alocare a resurselor la un nivel mult mai fin, astfel încât să existe o mai bună utilizare a acestora.

5.2.3 OPTIMIZARE CE ŢINE CONT DE TIPUL ENTITĂŢILOR PENTRU CARE SE EFECTUEAZĂ CALCULE

Ultima optimizare propusă vizează o alocare a resurselor care sa ţină cont şi de tipul de entitate pentru care se vor efectua calcule. Mai exact, se va face o diferenţiere între următoarele tipuri de entităţi ce populează spaţiul virtual :

- utilizatori umani
- utilizatori controlaţi de server : NPC (non player characters)

Într-un spaţiu virtual 3D MMO, cantitatea totală de NPC-uri raportată la numărul de utilizatori umani poate fi mai mare de zeci, chiar sute de ori. De asemenea, NPC-urile fiind controlate de către modulul de inteligenţă artificială a serverului, au un alt comportament decât cel al unui utilizator normal astfel şi operaţiile generate de acestea sunt diferite faţă de cele ale unui utilizator uman.

5.3 ABSTRACTIZAREA MATEMATICĂ A OPTIMIZĂRILOR PROPUSE

Pentru a putea folosi programatic în cadrul soluţiei arhitecturale optimizările propuse mai sus, se va folosi pentru determinarea unui cost asociat fiecărui task computaţional declanşat de către un utilizator, următoarea funcţie de cost :

$$Cost(task,utilizator) = PozitionareSpatiala(task,utilizator) * pondereSpatiala + ProfilingOperatii(task,utilizator) * pondereOperatii + TipEntitate(task,utilizator) * pondereTipEntitate$$

unde,

PozitionareSpatiala(task,utilizator), ProfilingOperatii(task,utilizator) şi *TipEntitate(task,utilizator)* sunt funcţii asociate optimizărilor prezentate anterior şi vor întoarce un cost determinat de acestea

iar,

pondereSpatiala, pondereOperatii şi *pondereTipEntitate* sunt ponderi asociate costurilor determinate de cele 3 funcţii care vor determina contribuţia acestora la costul total al task-ului

5.4 PREZENTAREA SOLUTIEI ARHITECTURALE COMPLETE

Pentru a putea introduce posibilitatea de execuţie de operaţii GPGPU în cadrul arhitecturilor pentru servere de spaţii virtuale 3D MMO a fost necesară introducerea/modificarea şi implementarea următoarelor componente :

- componenta de alocare a clienţilor şi resurselor
- componenta pentru creare / planificare sarcini :
    - global la nivel de server
    - specializat la nivel de GPU
        - *single GPGPU*
        - *multi GPGPU*
- componenta pentru procesarea şi propagarea rezultatului operaţiilor CUDA

Soluţia prezentată în continuare are drept scop obținerea unui nivel ridicat de scalabilitate, atât vertical cât şi orizontal, precum şi reducerea limitărilor arhitecturilor tradiţionale pentru servere de spaţii virtuale 3D MMO prin implementarea unei arhitecturi cu următoarele caracteristici importante :

- descarcă operaţiile intensiv computaţionale din sarcina CPU şi le implementează ca operaţii GPGPU;
- este proiectată sa fie masiv paralelă prin suportul planificării operaţiilor la nivel de multi GPGPU;
- foloseşte un model care este condus de evenimente şi sarcini, fiind centrat în principal pe locaţia, tipul şi acţiunile executate de entităţile utilizator în cadrul spaţiului virtual, deoarece ei sunt consumatorii puterii de calcul la nivel de server şi nu datele spaţiului virtual în sine.

5.4.1 MODULUL ALOCARE CLIENŢI ŞI RESURSE

Acest modul este responsabil cu alocarea clienţilor şi a resurselor necesare executării calculelor la nivel de server. Pentru realizarea alocării se folosesc criteriile prezentate in paragraful 5.2.

5.4.2 MODULUL PLANIFICARE SARCINI

Acest modul este responsabil cu crearea şi planificarea sarcinilor ce vor fi executate la nivelul serverelor spaţiilor virtuale 3D MMO. În arhitectura propusă, modulul de planificare a sarcinilor are două componente majore :

- componenta de creare a sarcinilor la nivel global (de sistem)
- componenta de planificare a sarcinilor la nivel de GPU / CPU

5.4.2.1 Creare sarcini la nivel de sistem

Această componentă realizează crearea de sarcini de calcul asociate operaţiilor executate în cadrul spaţiului virtual. Astfel, la fiecare iteraţie (*tick*) a buclei principale a simulării, această componentă procesează acţiunile entităţilor spaţiului virtual, creează sarcinile ce rezultă în urma procesării şi le inserează într-o coadă de sarcini globală. Tot această componentă este responsabilă şi cu asigurarea menţinerii ratei interne cu care funcţionează spaţiul virtual. De asemenea, tot această componentă realizează trimiterea sarcinilor pentru a fi planificate la componenta de planificare.

5.4.2.2 Planificare sarcini la nivel de GPU / CPU

Această componentă este responsabilă cu planificarea şi trimiterea spre execuţia propriu-zisă a sarcinilor la CPU sau la GPU. Pentru implementarea GPGPU, folosind costul asociat fiecărui task, acest modul realizează planificarea sarcinilor ce vor fi executate pe multiprocesoarele GPU din sistem astfel :

- determină numărul de resurse de calcul (GPU) din sistem;
- în funcţie de *tick-ul* intern si numărul de resurse de calcul, determină numărul maxim de sarcini ce pot fi executate la fiecare iteraţie a buclei principale a simulării spaţiului virtual;
- în funcţie de numărul de sarcini ce vor fi executate, determină dimensiunea blocurilor de fire de execuţie si a grid-ului ce le conţine;
- extrage din coada de sarcini globală acele sarcini ce vor fi planificate în *tick-ul* curent, în funcţie de costul asociat acestora;
- creează vectorul de sarcini de executat folosind sarcinile extrase la pasul anterior şi costul asociat acestora.

5.4.3 MODULUL DE PROCESARE CUDA

Acest modul primeşte sarcinile GPGPU de la modulul de planificare şi este responsabil cu execuţia acestora direct pe procesoarele plăcii grafice.

La nivelul acestui modul sunt realizate următoarele operaţii :

- implementarea funcţiilor kernel ce vor fi executate de firele de execuţie ale procesoarelor scalare ale GPU-ului
- transferul de memorie de la dispozitivul gazdă la GPU
- după ce sarcinile au fost executate pe GPU, se realizează transferul în sens invers al memoriei (ce conţine rezultatele execuţiilor) de la GPU la gazdă
- operaţiile de sincronizare necesare pentru asigurarea faptului ca toate firele de execuţie ale GPU-ului şi-au terminat procesările

Tot acest modul este responsabil şi cu propagarea rezultatelor execuţiilor către nivelul ce implementează logica spaţiului virtual.

6 TESTAREA SCALABILITĂŢII SOLUŢIEI

Pentru a realiza testarea de scalabilitate a fost realizat un prototip funcţional ce implementează funcţionarea unui spaţiu virtual 3D MMO bazat pe soluţia arhitecturală propusă în capitolul anterior. Astfel, au fost implementate module atât pentru partea de server cât şi pentru cea de client.

6.1 SPECIFICAŢII DE FUNCŢIONARE SERVER ŞI CLIENT

La nivelul arhitecturii serverului au fost implementate modulele descrise în capitolul anterior :

1. **Modulul de alocare a resurselor**
2. **Modulul de creare şi planificare a sarcinilor**
3. **Modulul de procesare a sarcinilor**

Serverul pune la dispoziţie următoarele funcţionalităţi către clienţii care accesează spaţiul virtual:

- Permitere conectare clienţi
- Autentificare clienţi
- Trimitere evenimente către clienţi :
- **Verificare corectitudine deplasare utilizator în spaţiul virtual**

Modulul client implementează următoarele funcţionalităţi importante :

- interacţiunea cu utilizatorul;
- comunicarea cu serverul folosind interfata de programare (API) pusă la dispoziţie de acesta;
- redarea 3D a spaţiului virtual.

Modulul client oferă clienţilor ce accesează spaţiul virtual următoarele:

- Conectare/Deconectare la spaţiul virtual;
- Autentificare prin username/parolă;
- Interacţiune cu utilizatorul
- Clientul funcţionează în următoarele 2 moduri
    - Mod vizualizare 3D a spaţiului virtual :

- Mod consolă : acest mod este necesar pentru a putea simula cu uşurinţă conectarea unui număr ridicat de utilizatori la spaţiul virtual

6.2 REZULTATE

Pentru a simula încărcarea spaţiului virtual cu un număr mare de clienţi a fost folosită aplicaţia client în modul de funcţionare consolă. Astfel, au fost pornite pentru simularea clienţilor mai multe instanţe ale acesteia, fiecare primind un fişier ce conţine comenzi de deplasare la poziţii aleatoare din spaţiul virtual cu scopul de a simula un comportament pentru fiecare client simulat.

Au fost realizate teste separate pentru următoarele numere de clienţi ce au fost simulaţi că accesează spaţiul virtual :

- 100 de clienţi, 1000 de clienţi, 4000 de clienţi şi 8000 de clienţi

Pentru efectuarea calculelor s-au testat toate cele 3 abordări :

- CPU : Intel Core2Quad Q6600
- Single GPGPU : 1 core al unui GPU GTX590
- Multi GPGPU : 4 core-uri GPU (2 GPU GTX590)

S-a avut în vedere timpul total necesar pentru calculul coliziunilor la fiecare iteraţie a buclei principale a spaţiului virtual. *Tick-ul* intern folosit în implementarea serverului este de 30 de milisecunde, deci practic orice depăşire a acestui prag conduce la imposibilitatea procesării în timp real pe partea de server a buclei principale de simulare, apărând întârzieri vizibile la nivelul aplicaţiei client.

Rezultatele sunt prezentate în continuare.

	100 de clienţi	1000 de clienţi	4000 de clienţi	8000 de clienţi
	Timp calcule fizică / iteraţie	Timp calcule fizică / iteraţie	Timp calcule fizică / iteraţie	Timp calcule fizică / iteraţie
1 CPU Quad Core Q6600	4.7ms	46ms	185ms	365ms
Single GPGPU 1 core GPU GTX590	2ms	7ms	23ms	42ms
Multi GPGPU 4 cores GPU GTX590	3ms	6.8ms	9ms	11ms

Tabel 6-1: Rezultate prototip funcţional al arhitecturii propuse

Din rezultatele obţinute se pot astfel observa următoarele :

- pentru 1000 de clienţi, implementarea pe CPU deja nu mai este în stare să facă faţă în timp real ratei interne la care funcţionează spaţiul virtual (30 de milisecunde);
- implementarea *single-GPU* are un speedup fata de cea CPU ce creşte o dată cu numărul de utilizatori şi de abia când numărul acestora atinge 8000 nu mai este în stare sa facă faţă la procesări în timp real;
- implementarea *multi-GPGPU*, datorită overhead-ului rezultat din operaţiile de transfer de memorie şi sincronizare, obţine sporul de performanţă aşteptat faţă de cea *single-GPGPU* doar atunci când numărul de utilizatori este unul ridicat; astfel, chiar şi pentru un număr de 8000 de utilizatori este capabilă să proceseze volumul de operaţii în timp real.

7 CONCLUZII ŞI CONTRIBUŢII ORIGINALE

Popularitatea din ce în ce mai mare a spaţiilor virtuale 3D MMO a condus inevitabil şi spre cresterea semnificativă a numărului utilizatorilor care accesează astfel de spaţii. Acest lucru a condus la următoarele două consecinţe majore :

- pentru a crea o experienţă cât mai bogată şi realistă pentru utilizatorii care accesează spaţiile virtuale, aplicaţiile MMO trebuie sa poată oferi la nivelul aplicaţiei client o redare 3D cât mai fidelă şi realistă din punct de vedere grafic;
- a împins spre extrem încărcarea cu care se confruntă arhitectura acestor aplicaţii la nivel de server, uneori aceasta nemaifăcând faţă numărului foarte mare de utilizatori.

A fost realizată cercetarea posibilităţilor de îmbunătăţire a tehnologiilor pentru spaţii virtuale MMO în următoarele direcţii majore:

- la nivelul redării 3D oferite utilizatorilor:
- la nivelul serverelor de spaţii virtuale 3D MMO

În continuare se va prezenta un rezumat al contribuţiilor originale aşa cum reies din capitolele anterioare.

7.1 CONTRIBUŢII ORIGINALE

➢ ANALIZA CONCEPTELOR, TEHNOLOGIILOR ŞI ARHITECTURILOR CURENTE FOLOSITE DE APLICAŢIILE MMO

A fost efectuată o amplă analiză a conceptelor şi tehnologiilor folosite pentru realizarea aplicaţiilor MMO atât la nivel de client cât şi de server. De asemenea, au fost analizate arhitecturile curente folosite în implementarea spaţiilor virtuale 3D MMO, punându-se în evidenţă atât avantajele utilizării acestora cât şi lipsurile cu care se confruntă. Această analiză, a dus la stabilirea unui punct de vedere propriu şi original asupra tehnologiilor folosite în domeniu, stând la baza cercetărilor ulterioare şi a soluţiilor propuse.

➢ UTILIZAREA TEHNICILOR GPGPU PENTRU REDAREA 3D LA CLIENT

Pentru a crea un grad de imersiune cât mai ridicat (prin creşterea realismului vizualizării lumii virtuale 3D) pentru utilizatorii spaţiului MMO, la nivelul aplicaţiei client se poate folosi ca metodă de redare algoritmul RayTracing. Odată cu lansarea la sfârşitul anului 2006 a arhitecturilor unificate pentru GPU, a apărut posibilitatea ca o gamă largă de operaţii de uz general sa poată fi implementate folosind tehnici GPGPU. Au fost astfel propuse soluţii pentru a executa algoritmul RayTracing folosind procesare *single-GPGPU* şi *multi-GPGPU*. Solutiile propuse au fost implementate şi au fost prezentate rezultatele obţinute. Este important de menţionat că la timpul realizării cercetării (2008 şi 2009) încercările de exploatare a paralelismului de tip GPGPU pentru implementarea algoritmului RayTracing se aflau într-o stare incipientă astfel încât cercetarea realizată în acel moment a fost una de actualitate.

➢ UTILIZAREA TEHNICILOR GPGPU ÎN CADRUL SERVERELOR MMO

La nivelul serverelor de spaţii virtuale 3D MMO se execută un număr foarte mare de operaţii pentru a simula o fizică realistă a spaţiului virtual (detecţia coliziunilor, forţe de frecare, forţa de gravitaţie, etc). A fost propusă o soluţie de paralelizare folosind procesare *single-GPGPU* şi *multi-GPGPU* a calculelor de detecţie a coliziunilor ce sunt executate în mod obişnuit în cadrul spaţiilor virtuale. Solutia a fost şi implementată.
Rezultatele obţinute în urma implementării soluţiei au confirmat validitatea ei şi au scos în evidenţă următoarele aspecte majore :

- implementarea folosind CPU nu mai face faţă în timp real pentru un număr relativ mediu de obiecte existente în scenă; se poate deduce astfel că procesarea pe CPU a calculelor de fizică nu este scalabilă cu numărul de obiecte din scenă;

- implementarea ce foloseşte single-GPGPU obţine sporuri de performanţă de ordinul zecilor faţă de CPU;

- implementarea ce foloseşte *multi-GPGPU* obţine sporuri de performanţă de ordinul sutelor faţă de CPU.

- **PROPUNEREA UNUI MECANISM DE ALOCARE DINAMICĂ A RESURSELOR, CENTRAT PE UTILIZATORII SPAŢIULUI VIRTUAL**

A fost propusă folosirea unui mecanism de alocare dinamică a resurselor necesare execuţiei calculelor, care este centrat pe utilizatorii spaţiului virtual deoarece aceştia generează necesarul computaţional şi nu datele lumii virtuale în sine. Soluția propusă vizează optimizarea alocării resurselor ținându-se cont de următorii 3 factori :

- poziţionarea spaţială a utilizatorilor care sunt on-line în cadrul spaţiului virtual;
- un cost de „greutate a operaţiilor efectuate” asociat fiecărui utilizator care este on-line în cadrul spaţiului virtual;
- tipurile de entităţi pentru care trebuie realizate calcule.

- **SOLUŢIE ARHITECTURALĂ COMPLETĂ CU POSIBILITATEA EXPLOATĂRII PARALELISMULUI DE TIP GPGPU**

A fost propusă o soluţie arhitecturală completă ce conţine modulele necesare integrării în arhitectura serverelor de spaţii virtuale 3D MMO a soluţiilor de optimizare prezentate

- **IMPLEMENTAREA UNUI PROTOTIP FUNCȚIONAL PENTRU ARHITECTURA PROPUSĂ ŞI TESTAREA DE SCALABILITATE A ACESTEIA**

A fost implementat un prototip funcţional al arhitecturii propuse, ce foloseşte:

- procesare GPGPU a sarcinilor aferente operaţiilor de detecţie a coliziunilor;
- alocarea dinamică a resurselor GPU în funcţie de poziţionarea spaţiala a acestora în cadrul lumii virtuale.

A fost implementată aplicaţia server, aplicaţia client precum şi un mecanism de simulare a unui număr mare de clienţi ce accesează spaţiul virtual pentru a se putea realiza testele de scalabilitate. Rezultatele obţinute au confirmat validitatea şi viabilitatea soluţiei arhitecturale propuse.

8 LISTA LUCRĂRILOR ŞTIINŢIFICE ALE AUTORULUI

8.1 LUCRĂRI LEGATE DE TEMATICA TEZEI

8.1.1 CĂRŢI

[MMA09a] Alin Moldoveanu, Florica Moldoveanu, **Victor Asavei**, Costin Boiangiu
Realitatea Virtuală
Editura Matrix Rom, ISBN:978-973-755-488-8, 216 pg., 2009

8.1.2 ARTICOLE

[AMM09a] **Victor Asavei**, Florica Moldoveanu, Alin Moldoveanu
Ray Tracing as GPGPU
CSCS 17 - The 17th International Conference On Control Systems And Computer Science, 26-29 May 2009, Bucharest

[AMM09b] ***Victor Asavei***, *Vlad-Valentin Ionita, Florica Moldoveanu, Alin Moldoveanu*
Real-Time Parallel Volume Rendering using Nvidia Cuda for Medical Imaging
1483-1485, Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium, ISBN 978-3-901509-70-4, ISSN 1726-9679, pp 742, Editor B[ranko] Katalinic, Published by DAAAM International, Vienna, Austria 2009 **(Proceeedings ISI)**

[AMM09c] ***Victor Asavei***, *Florica Moldoveanu, Alin Moldoveanu, Costin-Anton Boiangiu, Ruxandra Marasescu*
Ray Tracing as Multi GPGPU
1203-1205,Annals of DAAAM for 2009 & Proceedings of the 20th International DAAAM Symposium, ISBN 978-3-901509-70-4, ISSN 1726-9679, pp 602, Editor B[ranko] Katalinic, Published by DAAAM International, Vienna,Austria 2009 **(Proceeedings ISI)**

[AMM10a] ***Victor Asavei***, *Alin Moldoveanu, Florica Moldoveanu, Anca Morar, Alexandru Egner*
GPGPU for Cheaper 3D MMO Servers

International Conference on Telecommunications and Informatics : Session Information Science and Applications, Catania, Italia, 29-31 Mai 2010 **(Proceeedings ISI)**

[AMB10] ***Asavei, V.;*** *Moldoveanu, A. D. B.; Moldoveanu, F.; Boiangiu, C. A.;Morar, A. - A. & Egner, A. I.*

Innovative 3D MMO Servers Architectures Based on GPGPU

Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, ISBN 978-3-901509-73-5, ISSN 1726-9679, pp 0325, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2010 **(Proceeedings ISI)**

[AMM11] ***Victor Asavei****, Florica Moldoveanu, Alin Moldoveanu, Alexandru Egner, Anca Morar*

Multi GPGPU Optimizations for 3D MMO Virtual Spaces

CSCS 18 - The 18th International Conference On Control Systems And Computer Science, 24-27 May 2011, Bucharest

[MMA08a] *Alin Moldoveanu, Florica Moldoveanu, Alexandru Soceanu,* ***Victor Asavei***

A 3D Virtual Museum

Scientific Bulletin of UPB, Series C, vol. 70, No.3, ISSN 1454-234x, 2008

[MMA08b] *Alin Moldoveanu, Florica Moldoveanu,* ***Victor Asavei***

Méga Musée Virtuel

Conférence Internationale Francophone d'Automatique (CIFA) , ISBN: 978-2-915913-24-8 , Bucarest, 3-5 Septembre 2008

[MMA09b] *Alin Moldoveanu, Florica Moldoveanu,* ***Victor Asavei***

Highly Scalable Server Architecture for Massive Multi-player 3D Virtual Spaces

"Recent Advances în Applied Mathematics and Computaţional and Information Sciences", vol II, Proceedings of the 15th American Conference on Applied Mathematics and Proceedings of the International Conference on Computaţional and Information Science 2009, Houston, USA, April 30-May 2, 2009, ISSN: 1790-5117, ISBN: 978-960-474-071-0

[MMA09c] *Alin Moldoveanu, Florica Moldoveanu,* ***Victor Asavei***

More scalability at lower costs – Server Architecture for Massive Multi-player 3D Virtual Spaces powered by GPGPU

International Journal of Computers and Communication, Issue 2, Volume 1, 2009, ISSN: 2074-1294, published by University Press, London, UK, 2009

[MMA09d] *Alin Moldoveanu, Florica Moldoveanu,* ***Victor Asavei***

Méga Musée Virtuel

Automatique Avancee et Informatique Appliquee, Editura Academiei Romane, 2009

[MMA11] *Alin Dragos Bogdan Moldoveanu, Florica Moldoveanu,* ***Victor Asavei****, Alexandru Egner, Anca Morar*

From HTML to 3DMMO - a Roadmap Full of Challanges

CSCS 18 - The 18th International Conference On Control Systems And Computer Science, 24-27 May 2011, Bucharest

[LMM11] *Lucian Petrescu, Anca Morar, Florica Moldoveanu and* ***Victor Asavei***

Real Time Reconstruction of Volumes from Very Large Datasets Using CUDA

International Conference on System Theory, Control and Computing, 14-16 October, Sinaia, Romania (accepted as a full paper and for publication în the conference proceedings)

8.2 ALTE LUCRĂRI REALIZATE ÎN PERIOADA DESFĂŞURĂRII DOCTORATULUI

[MMM10a] *Anca Morar, Florica Moldoveanu, Alin Moldoveanu,* ***Victor Asavei****, Alexandru Egner*

Computer Assisted Analysis of Orthopedic Radiographic Images

International Conference on Signal Processing : Session Image Processing, Catania, Italia, 29-31 Mai 2010 **(Proceeedings ISI)**

[MMM10b] *Anca Morar, Florica Moldoveanu, Alin Moldoveanu,* ***Victor Asavei****, Alexandru Egner*

Medical Image Processing in Hip Arthroplasty

WSEAS TRANSACTIONS on SIGNAL PROCESSING, Volume 6, Print ISSN: 1790-5052, E-ISSN: 2224-3488, 2010

[MMM10c] *Anca Morar, Florica Moldoveanu, Alin Moldoveanu,* ***Victor Asavei***

Computer Assisted Analysis of 2D/3D Medical Images

Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, ISBN 978-3-901509-73-5, ISSN 1726-9679, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2010 **(Proceeedings ISI)**

[EMM10] Egner*, A. I.; Moldoveanu, F.; Moldoveanu, A. D. B.;* ***Asavei, V****.; Morar, A. - A.; & Boiangiu, C. A*.

Testing the interoperability of HL7-based applications using TTCN-3

Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, ISBN 978-3-901509-73-5, ISSN 1726-9679, Editor B. Katalinic,Published by DAAAM International, Vienna, Austria 2010 **(Proceeedings ISI)**

[MAM10] *Moldoveanu, A. D. B.;* ***Asavei, V****.; Moldoveanu, F; Morar, A. - A. & Egner, A. I. .; Boiangiu, C. A.*

Turning Nearshoring into a Success – Managing Technical Background Differences
Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, ISBN 978-3-901509-73-5, ISSN 1726-9679, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria 2010 **(Proceeedings ISI)**

[BPM10] *Boiangiu, C. A.; Petrescu, S.B. ; Moldoveanu, A. D. B.; .;* ***Asavei, V.****; Bucur, I.*

Systematic Printing Space Recognition
Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Editor B. Katalinic,Published by DAAAM International, Vienna, Austria 2010 **(Proceeedings ISI)**

[APM11] ***Victor Asavei****, Alexandru-Lucian Petrescu, Florica Moldoveanu*

A Fractal Approach to Terrain Generation and Rendering
CSCS 18 - The 18th International Conference On Control Systems And Computer Science, 24-27 May 2011, Bucharest

[EMM11] *Alexandru Egner, Florica Moldoveanu, Alin Moldoveanu, Victor Asavei, Anca Morar*

Automated generation of TTCN-3 type set used for testing of healthcare applications
CSCS 18 - The 18th International Conference On Control Systems And Computer Science, 24-27 May 2011, Bucharest

[MMM11] *Anca Morar, Florica Moldoveanu, Alin Moldoveanu, Victor Asavei, Alexandru Egner*

Computer Assisted Insertion of Prostheses Based on Medical Images
CSCS 18 - The 18th International Conference On Control Systems And Computer Science, 24-27 May 2011, Bucharest

9 BIBLIOGRAFIE

[MIL10] John L. Miller, John Crowcroft
The near-term feasibility of P2P MMOG's
NetGames '10 Proceedings of the 9th Annual Workshop on Network and Systems Support for Games

[LAK09] Geetika T. Lakshmanan, Yuri G. Rabinovich, Opher Etzion
A stratified approach for supporting high throughput event processing applications
Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, 2009

[MAR09] Alma Martinez G., Hector Orozco A., Felix Ramos C., Mario Siller
A Peer-To-Peer Architecture for Real-Time Distributed Visualization of 3DCollaborative Virtual Environments
2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications

[CHE08] Che S. , Others
A Performance Study of General Purpose Applications on Graphics Processors
Journal of Parallel and Distributed Computing, Volume 68, Issue 10, Pages 1370-1380, 2008

[NVI09a] NVidia Corporation
NVIDIA OptiX RayTracing Engine,
SIGGRAPH 2009, NEW ORLEANS—Aug. 4, 2009

[CAR11] John Carmack,
GPU Race, Intel Graphics, Ray Tracing, Voxels and more!,
HTTP://WWW.PCPER.COM/REVIEWS/EDITORIAL/JOHN-CARMACK-INTERVIEW-GPU-RACE-INTEL-GRAPHICS-RAY-TRACING-VOXELS-AND-MORE

[BRE08] Breitbart J.
Case studies on GPU usage and data structure deşign
Dept. of Computer Science and Electrical Engineering, Universitat Kassel, 2008

[FOL05] Foley T. and Sugerman J.
Kd-tree acceleration structures for a GPU Raytracer
ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, ACM Press, Pages 15-22

[CHE06] Alvin Chen and Richard R. Muntz
Peer Clustering: A Hybrid Approach to Distributed Virtual Environments

The 5th Workshop on Network & System Support for Games 2006 — NETGAMES 2006

[WAL04] Wald, Ingo
Real Time Ray-Tracing and Interactive Global Illumination
2004

[MOL96] Moldoveanu, Florica & Alţii
Grafica pe calculator
Ed. Teora, Bucureşti, 1996

[QUA04] Quake3 RayTraced
HTTP://WWW.Q3RT.DE

[QUA06] Quake4 RayTraced
HTTP://WWW.Q4RT.DE

[QWR08] Quake Wars RayTraced
HTTP://WWW.QWRT.DE

[WOL10] Wolfenstein RayTraced
HTTP://WWW.WOLFRT.DE

[GAI11] Gaikai
HTTP://WWW.GAIKAI.COM

[ONL10] OnLive
HTTP://WWW.ONLIVE.COM

[EVE11] Eve OnLine
HTTP://WWW.EVEONLINE.COM

[WOW11] World of Warcraft
HTTP://WWW.WOW-EUROPE.COM

[CUD11] CUDA Programming Guide
HTTP://WWW.NVIDIA.COM/OBJECT/CUDA_DEVELOP.HTML

[FAT08] Kayvon Fatahalian and Mike Houston
A Closer look at GPUs
Communication of the ACM, vol 51, no 10, 2008

[CHA06] Chris Chambers, Wu-chang Feng, Wu-chi Feng
Towards public server MMOs

Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, 2006

[KES05] J. Kesselman

Server Architectures for Massively Multiplayer Online Games

În Session TS-1084, Javaone conference, SUN, 2005

[WIE06] A. Wierzbicki

Trust Enforcement în Peer-to-peer Massive Multi-player Online Games

Proc. Grid computing, high-performAnce and Distributed Applications (GADA'06), Springer, LNCS 4276 (2006), 1163-1180

[GUP09] Nitin Gupta and others

Scalability for Virtual Worlds

Proceedings ICDE '09 Proceedings of the 2009 IEEE International Conference on Data Engineering

[BEZ08] Carlos Eduardo B. Bezerra, F´abio R. Cecin, Cl´audio F. R. Geyer

A3: a Novel Interest Management Algorithm for Distributed Simulations of MMOGs

12th 2008 IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications

[GLI08] Frank Glinka, Alexander Ploss, Sergei Gorlatch, and JensM¨ uller-Iden

High-level development of multiserver online games

International Journal of Computer Games Technology - Networking for Computer Games, Volume 2008, January 2008