University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

Distribuovaný výpočet digitálních hologramů

Technical report

Martin Janda, Ivo Hanák, Václav Skala

Technical Report No. DCSE/TR-2007-12
Decemeber, 2007

Distribution: public

Technical Report No. DCSE/TR-2007-12
Decemeber, 2007

Distribuovaný výpočet digitálních hologramů

Martin Janda, Ivo Hanák, Václav Skala

Abstract

Tento dokument je zprávou o řešení aktivity 7ZHO10.

This work has been partialy supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics). This work has been partialy supported by the EU project EU within FP6 under Grant 511568 with the acronym 3DTV

Copies of this report are available on
`http://www.kiv.zcu.cz/publications/`
or by surface mail on request sent to the following address:

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitni 8
30614 Pilsen
Czech Republic

Authors hereby declare that this is their own work and all materials are properly cited.

Obsah

Kapitola 1

Úvod

Dokument popisuje naše experimenty s distribuovanými výpočetními systémy z pohledu syntézy hologramu. Předložena je krátká analýza a popis vyzkoušených řešení. Na závěr je pak provedeno zhodnocení výsledku a vyvozené závěry.

Cílem našeho snažení je vyvinout metodu, která by byla schopna počítat velké hologramy libovolných scén. Zásadní problémem, který je však jen obtížně řešitelný, je algoritmická složitost syntézy, která je $O(N^2M)$, kde N^2 je počet vzorků výsledného optického pole a M je počet vzorků scény, který je srovnatelný s počtem vzorků, tj. $M \sim N^2$.

Uvedená skutečnost tedy znamená, že faktická algoritmická složitost je srovnatelná s $O(N^4)$. Narozdíl od [KZG07] není naším cílem aplikovat principy vlnového šíření světla na počítačovou grafiku se záměrem zvýšit vizuální věrnost obrazu. Naším cílem je holografie pro holografické displeje a proto není v našem případě $N \sim 1024$, ale minimálně $N \sim 4096$ a ideálně by mělo být N v řádech desetitisíců. Výpočet takto velikého optického pole je jen velmi obtížně zvládnutelný na běžném počítači.

Algoritmická redukce na problém nižší složitosti je sice možná, ale pouze pro velmi speciální případ dvou rovnoběžných rovin. V obecném případě není dosud nalezen postup, který by umožnil redukci složitosti zásadním způsobem, tj. min. o jeden řád. Jako jediné řešení se v tuto chvíli jeví nalézt takovou interpretaci algoritmu syntézy, která by umožnila paralelní nebo distribuovaný výpočet s minimální, ideálně žádnou, sekvenční částí. I když tento přístup bude v nejlepším případě znamenat pouze lineární urychlení, při dosahovaných výpočetních časech, se bude jednat o urychlení znatelné.

Před zahájením hledání technologických prostředků pro řešení úlohy jsme vyloučili paralelní zpracování, neboť předpokládá běh na jednom stroji. Pomineme-li skutečnost, že stroje s větším množství procesorů (¿ 8) jsou obtížně dostupné, multiprocesorový stroj je stále jen stroj se sdílenou pamětí. Předpokládáme-li, že cílem metody je hologram přesahující velikost běžně dostupné paměti ($>$ 4 GB), pak sdílená paměť znamená zavedení odkládacího mechanismu, který může znehodnotit veškeré urychlení získané paralelizací.

V našem řešení jsme se tedy soustředili na distribuovaný výpočet. Základní výhodou distribuovaného výpočtu z pohledu digitální holografie je snadná dostupnost. Díky tomu, že každý výpočetní uzel zpracovává jen jeden úsek celé úlohy, je mechanismus odkládání výpočetně i paměťově jednodušší. Dále, vzhledem ke konstrukci naší metody je možné sestavit libovolně velký výpočetní svazek počítačů a získat tak libovolný, ač lineární koeficient urychlení.

Kapitola 2

Technické a programové prostředky

Kapitola předkládá prostředky, ať již programové nebo technické, které byly zvažovány a vyzkoušeny při řešení úlohy. Každý z prostředků je opatřeny krátkým popisem, použité prostředky pak popisem detailnějším.

2.1 Prostředky platformy .NET

Platforma .NET byla jako výchozí platforma zvolena především proto, že v době prvních pokusů s distribuovaným výpočtem byla drtivá většina kódu implementující první verze našeho algoritmu [JHS06] napsána právě pro prostředí .NET. A tedy, prvním z prostředků, který byl zvažován je mechanismus Remoting [CLR] poskytovaný platformou .NET.

Mechanismus Remoting umožňuje spouštět zdrojový kód na vzdáleném stroji, serveru. Klient, který spouští kód na vzdáleném stroji, pracuje s tímto vzdáleným objektem podobně jako s objektem lokálním, což velmi zjednodušuje implementaci algoritmu. Prostředí .NET zajišťuje vzdálenou komunikaci, vzdálené volání metody a správu vzdálených objektů.

Nevýhodou mechanismu je rychlost komunikace, která je zatížena vlastní správou skrytou před uživatelem. V běžném použití pro potřeby vzdálených služeb databázových strojů je tato vlastnost žádoucí. Pokud však je plánováno, že vzdálený objekt bude předávat a zpracovávat velké kusy dat, lze předpokládat, že vlivem obecnosti řešení bude přenos zatížen velkou režií. V nejhorším případě pak mohou být původně binární data překódována do textové podoby XML formátu. Navíc, lze předpokládat, že zvýšenou režií nebude zatížen pouze přenosový kanál, ale také paměť cílového a zdrojového stroje jako důsledek vyšší bezpečnosti běhu aplikace.

Další nevýhodou je pak nutnost ručně distribuovat kód, který bude vykonáván. Obecně .NET neposkytuje prostředky pro vzdálenou distribuci kódu určeného pro běh na vzdáleném stroji, což ztěžuje ladění výpočtu o více uzlech. V neposlední řadě, mechanismus Remoting spíše předpokládá, že existuje více klientů, kteří využívají služby silného serveru. Naše použití by znamenalo situaci zcela opačnou, 1 klient, který využívá služeb velkého množství klientů.

Na základě výše uvedených nevýhod nebyly standardní prostředky systému .NET pro

řešení a ani pro testovací implementaci použity. Namísto toho, byly prozkoumány možnosti knihoven poskytující efektivnější způsob spouštění vzdáleného kódu včetně jeho distribuce a distribuce dat.

2.2 SW knihovny pro distribuovaný výpočet

Vzhledem k tomu, že distribuovaný výpočet je pro naše účely prostředkem, nikoliv cílem, rozhodli jsme se použít nějaký již existující systém. Naše počáteční požadavky na tento systém byly:

- nativní podpora .NET,
- automatická distribuce kódu a dat,
- jednoduchost implementace.

Tyto podmínky splňovalo jen několik málo systémů. My jsme se zabývali:

- Alchemi: `http://www.alchemi.net/`,
- nGrid: `http://ngrid.sourceforge.net/`.

Naše zkušenosti těmito systémy jsou uvedeny v sekcích níže, v podstatě lze však říci, že v konečné fázi vždy převážily jejich nevýhody nad poskytovanými službami a tudíž nebyly zvoleny pro další implementaci a použití. Toto bylo jedním z důvodů, proč jsme se rozhodli přejít pod ANSI C++ a získat tak přístup k širší skupině prostředků, např. MPI.

2.2.1 Alchemi

První pokusy s distribuovaným výpočtem jsme prováděli s knihovnou Alchemi. Knihovna předpokládá heterogenní síť a nespolehlivé prostředí, tj. poskytuje prostředky pro detekci výpadku výpočetního uzlu. Knihovna poskytuje API pro tvorbu distribuované aplikace, nástroje pro sledování stavu výpočetní sítě a správu úloh. Hlavní součástí knihovny jsou i dvě klíčové aplikace Alchemi Manager a Alchemi Executor. Alchemi Manager je aplikace, která běží na řídícím uzlu a řídí celý distribuovaný výpočet. Alchemi Executor je aplikace, která běží na výpočetních uzlech a je zodpovědná za spouštění distribuovaných úloh, které jim jsou zadány řídícím uzlem.

Základní API knihovny je navrženo velice elegantně. Stěžejní součástí je speciální třída `GThread`, která má podobné rozhraní jako standardní třída určená pro práci s vláknem, tj. `Thread`. Z pohledu programátora je tedy možné konstruovat distribuovanou úlohu podobně jako paralelní aplikace s více vlákny a vlastní distribuce je pak zařízena systémem Alchemi. Výhodné na poskytnutém API je to, že není nutné fyzicky oddělovat kód na kód řídící výpočet, tj. farmer, a kód zpracovávající, tj. worker, jak je tomu například v knihovně PVM.

Systém Alchemi byl v době pokusů pořád ještě ve vývoji a trpěl tedy různými nedostatky a to především nestabilitou. Docházelo k výpadkům spojení mezi Alchemi Manager a Alchemi Executor. Také ve většině případů náš výpočet nedoběhl a došlo k pádu kódu

vykonávaného na jednotlivých uzlech či dokonce docházelo k pádu řídícího uzlu při pokusu o přenos výsledku. Především posledně jmenované chování nebylo odstraněno ani v následujících verzích, které spíše přidávaly další funkce, např. podporu .NET 2.0. Vzhledem k rozsahu zdrojových textů a času vyměřeného na provedení experimentů, jsme zavrhli snahu opravit tyto chyby vlastními silami. Uvedené nevýhody pak zcela knihovnu Alchemi vyloučili z dalšího zkoumání a pokusů.

2.2.2 nGrid

Další uvažovanou knihovnou pro distribuovaný výpočet byla knihovna nGrid. Tato knihovna je také určena pro prostředí .NET. Výhodnou knihovny nGrid oproti Alchemi je jednoduchost a přehlednost zdrojových kódu. Knihovna předpokládá, že pro její použití se provede úprava a doplnění jejího kódu. Nicméně vzhledem k nedobré zkušenosti s nedokončeným systémem Alchemi jsme od doplňování distribučního kódu od knihovny nGrid ustoupili a provedli konverzi našeho programového vybaveni z prostředí .NET do nativního kódu.

Uvedený krok byl sice časově náročný, avšak srovnatelný s dobou, kterou bychom museli investovat do úpravy knihovny nGrid a restrukturalizací našeho kódu vlivem této úpravy. Konverze do nativního kód nám umožnila využít odladěné a funkční knihovny MPI poskytované na strojích projektu METACentrum. Jelikož je na strojích projektu METACentrum použit operační systém Unix nebo jeho klony, byl pro převod do nativního kódu užit jazyk ANSI C++.

2.2.3 MPI

Vzhledem k nevýhodám a komplikacím s nástroji dostupnými pro prostředí .NET, jsme přešli k jazyku C++ a distribuovanému výpočtu s podporou MPI. Zásadní vlastností MPI je skutečnost, že se standardně předpokládá homogenní prostředí a není tedy nutné fyzicky oddělovat kód pro řídící uzel a výpočetní uzel, jak je tomu v případě PVM. Uvedená vlastnost také výrazně zjednodušuje vývoj aplikace. Dále pak vlivem předpokladu homogenity sítě je možné zjednodušit návrh algoritmu a snížit tak režii řízení výpočtu. V neposlední řadě se jedná o systém standardizovaný a přenositelný mezi platformami na úrovni zdrojového kódu a lze tedy očekávat, že výsledná aplikace využívající MPI bude bez větších úprav přeložitelná a spustitelná na rozličných výpočetních systémem. Homogenní síť výpočetních strojů je dostupná v rámci projekt METACentrum. Na základě výše vypsaných vlastností byla knihovna MPI zvolena pro implementaci této i našich dalších metod.

2.3 Technické vybavení pro distribuovaný výpočet

Distribuovaný výpočet se vyznačuje tím, že výpočet je prováděn na více než jednom stroji. Katedra informatiky a výpočetní techniky disponuje výpočetním blokem, který se skládá z 10 počítačů Intel Xeon 3.2 GHz, 2GB RAM, 80 GB HDD propojených 1 Gbps Ethernet sítí. Jednotlivé výpočetní uzly jsou spravovány operačním systémem Linux. Podobné vybavení je také využíváno v rámci projektu METACentrum, v němž se sdružují výpočetní stroje z několika univerzit viz níže.

2.3.1 METACentrum

Projekt na svých stránkách `http://meta.cesnet.cz/` o sobě píše:

Projekt METACentrum zastřešuje většinu aktivit souvisejících v České republice s GRIDy a/nebo výkonným počítáním obecně. METACentrum je jednou z významných aktivit výzkumného záměru sdružení CESNET.

Projekt METACentrum rozšiřuje infrastrukturu akademické vysokorychlostní sítě o podporu aplikací, vyžadujících rozsáhlé výpočetní kapacity. Vlastním cílem projektu METACentrum je propojení stávající výpočetní kapacity největších akademických center České republiky a její další rozšiřování.

METACentrum je aktuálně složeno z řady spolupracujících superpočítačových center sdílících výpočetní kapacity. V současnosti jsou pod vedením CESNETu zapojena následující klíčová pracoviště:

- Superpočítačové centrum Brno (Masarykova univerzita Brno - MU)
- Západočeské superpočítačové centrum (Západočeská univerzita Plzeň - ZČU)
- Superpočítačové centrum UK (Karlova univerzita Praha - UK)

Systémy projektem spravované vytváří virtuální distribuovaný počítač, v poslední době označovaný jako Grid. Smyslem aktivit je jednak odstínit uživatele od nepodstatných rozdílů mezi jednotlivými konkrétními systémy, tvořícími Grid, jednak umožnit jejich synchronní využití a poskytnout tak výpočetní kapacitu přesahující možnosti jednotlivých center.

Kapitola 3

Distribuovaná syntéza digitálních hologramů

V této kapitole se budeme věnovat možným uspořádáním algoritmu pro syntézu digitálního hologram v kontextu distribuovaného výpočtu a zvoleného výpočetního prostředí, tj. MPI. Kapitola předpokládá možná řešení distribuce výpočet uvádí také zjištěné výhody a nevýhody.

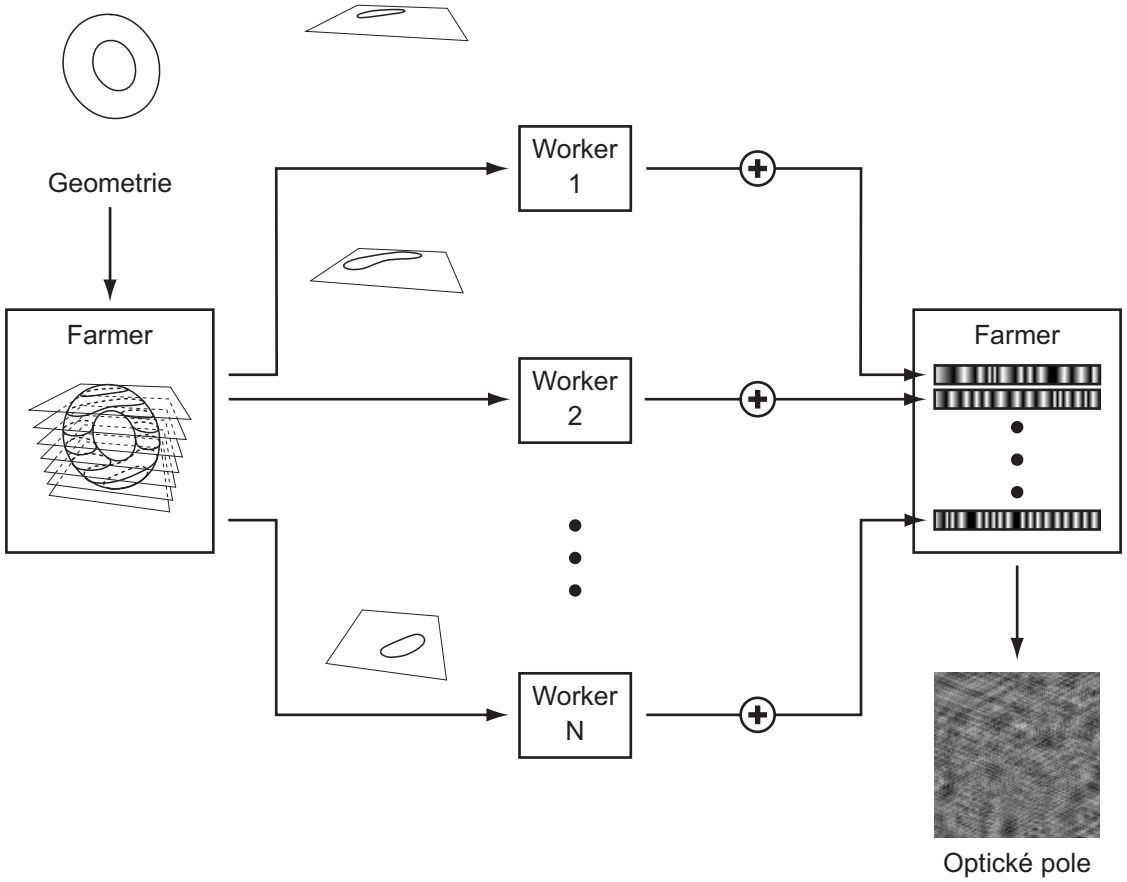
Zásadním problémem v distribuovaném výpočtu, který je nutné vyřešit, je rozdělení aplikace na souběžně běžící výpočty a to tak, aby vzájemná komunikace a výměna dat byla co nejnižší. Organizace metody pro syntézu digitálních hologramů [JHS07] tohoto rozdělení umožňuje. Výstupem metody je optické pole $U = [u_{p,q}]$. Jeho hodnota pole je součet optickým polí jednotlivých elementů objektů.

Metoda syntézy je založena na vrhání paprsků ze vzorku optického pole na pozici $x_{p,q}$ směrem do scény. Pro každý paprsek, který protne geometrii scény se z nejbližšího průsečíku vypočítá příspěvek, který je pak přičten k výsledné hodnotě $u_{p,q}$. Vzhledem k tomu, že vzorky $x_{p,q}$ leží na pravidelné a pravoúhlé mřížce, zvolili jsme postup vrhání paprsků, který je rozšířením způsobu popsaném v [JHS06].

V publikaci [JHS06] je popsán postup pro výpočet optického pole mající pouze horizontální paralaxu (HPO). V podstatě se jedná o 1D optické pole rovinných řezů geometrie, naskládaných na sobě. Geometrie scény se tedy postupně řeže rovinou rovnoběžnou s rovinou $\rho : z = 0$. Z každého řezu se spočítá 1D optické pole, viz [JHS06], který pak reprezentuje jeden řádek výsledného optického pole. Pokud bychom chtěli spočítat plnohodnotné optické pole s paralaxou v obou směrech, pak je nutné spočítat řadu HPO optických polí a ty nakonec sečíst. Každé HPO optické pole je složeno z 1D optických polí vzniklých z řezů rovinami, které jsou nakloněné o určitý úhel.

3.1 Organizace výpočtu

První způsob distribuce, který jsme testovali bylo využití modelu Farmář-Dělník. Farmář je zodpovědný za načítání geometrie scény a vytváření jejích řezů. Každý takový řez pak odešle právě volnému Dělníkovi. Ten z řezu spočítá 1D optické pole a to odešle zpět Farmáři. Ten 1D optické pole převezme a přičte k příslušnému řádku výsledného optického pole. Toto se opakuje dokud nejsou zpracovány všechny řezy. Mechanismus je ilustrován na obr. 3.1.



Obrázek 3.1: Distribuce syntézy digitálního hologramu metodou Farmář-Dělník. Farmář generuje řezy scény, které posílá ke zpracování dělníkům.

Nevýhoda tohoto řešení, která má zásadní vliv na celý výkon, je souběh předání výsledků při použití homogenních výpočetních uzlů. Pokud je velikost ortogonálního průmětu scény na snímací rovině srovnatelná s velikostí počítaného optického pole, pak lze předpokládat, že ve výpočtu budou existovat úseky s výpočetní dobou velmi podobnou či shodnou. V takové situaci pak požadavek a výsledky z předchozího kroku dojdou na stroj Farmáře v přibližně stejnou dobu což bude mít za následek prostoje jednotlivých výpočetních uzlů, neboť Farmář je schopen zpracovávat přicházející požadavky pouze sekvenčně. Redukce tohoto jevu rozdělením na větší úseky pak v sobě nese riziko, že se sice rozdíly mezi úseky výrazně zvětší, ale zároveň ke konci může z celého výpočetního uskupení uzlů počítat pouze jeden výpočetní uzel, neboť ostatní již své úlohy dokončily.

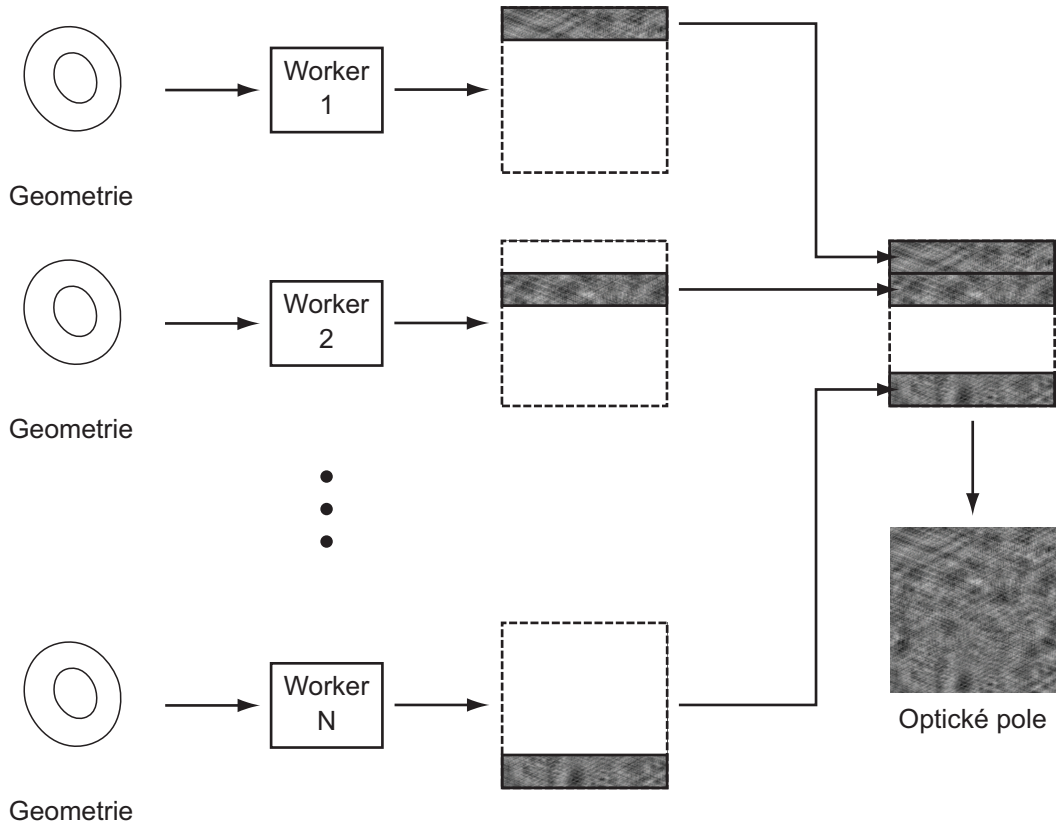
Výše uvedené nevýhody jsou samozřejmě řešitelné vytvořením sofistikovaného systém pro rozvrhování práce, který by byl schopen dynamicky odebírat a přidělovat práci a případně by byl schopen pozdržet odeslání výsledku na cílový výpočetní uzel. Nicméně časová náročnost implementace takového řešení by byla vysoká a tudíž k ní nebylo přistoupeno.

Pro účely naší úlohy bylo zvoleno řešení jednodušší, založené na skutečnosti, že cílové výpočetní uskupení strojů je homogenní a je postavené na rozhraní MPI. Tedy je možné dopředu rozvrhnout činnost a zaměstnat tak všechny výpočetní uzly bez nutnosti specializace na Farmáře a Dělníky.

Na základě zkušeností s první implementací se přešlo k organizaci výpočtu, která

připomíná výpočet organizovaný jako Single Program Multiple Data (SPMD). Všechny uzly výpočetní skupiny provádí stejný program, ale provádí ho nad jinou částí výsledného optického pole. V tomto případě při zahájení výpočtu může dojít ke komunikaci, při které si výpočetní uzly přidělí sekce optického pole, které budou vyhodnocovat. Sekce se skládají z jednoho či více řádků v závislosti na velikosti dostupné paměti. Po přidělení sekcí pak již uzel funguje samostatně. Načte si vlastní kopii scény a provádí si vlastní řezání. Vzhledem k homogenitě prostředí je možné přidělení provést s využití deterministického přístupu a pseudonáhodných čísel bez nutnosti vzájemné komunikace.

Poté co je výpočet dokončen ve všech sekcí na všech uzlech, odešlou se výsledné sekce optického pole do určeného adresáře a výpočet končí. Výsledné optické pole je kombinováno skládáním jednotlivých sekcí za sebe, čímž je možné snadno složit i optické pole přesahující dostupnou operační paměť. Vzhledem k tomu, že výpočet sekcí by měl trvat přibližně stejně dlouho, není třeba žádné řízení zátěže. V průběhu výpočtu tedy neprobíhá žádná komunikace. Tento způsob distribuce syntézy digitálních hologramů je ilustrován na obr. 3.2.



Obrázek 3.2: Distribuce syntézy digitálního hologramu metodou SPMD. Všechny uzly provádí stejný program - syntézu - ale jiné části optického pole.

Výhodou tohoto systému je minimální komunikace. V průběhu vlastního výpočtu uzly nekomunikují vůbec. Tím, že se jednotlivé sekce hologramu počítají přímo, nedochází tedy ke zbytečným přenosům. Toto ušetření je signifikantní zvlášť pro větší hologramy. Dále tím, že jednotlivé sekce jsou zvoleny tak, aby se vešly celé do operační paměti, výpočet není zpomalován nízkou zápisovou rychlostí externí paměti, kam by se jinak musel mezivýsledek odkládat. Tyto výhody značně převažují nevýhody, které jsou:

- Opakované zpracování geometrie, které se provádí při každé změně vertikálního úhlu naklonění řezné roviny.
- Nutnost kopírovat všechna data nutná pro popis scénu na jednotlivé výpočetní uzly před spuštěním výpočtu.

3.2 Možnosti vzdáleného výpočtu na GPU

Naše metoda se ukázala býti vhodná pro urychlení implementací na GPU [HJS07] a to i v situaci, že velikost cílového optického pole přesahuje maximální velikost textury [Mic07, SA06]. V praxi to znamená, že je možné implementovat výpočet na GPU s tím, že vypočtené bloky jsou odkládány do fyzické či externí paměti a paměť GPU je tak uvolněna pro další výpočet. Takováto struktura implementace je připravena pro distribuované nebo paralelní prostředí a zbývá pouze zakomponovat mechanismus distribuce aplikace a vzdáleného spouštění.

Zásadním problémem, které je nutné řešit, je možnost vzdáleného provedení výpočtu. Vzhledem k tomu, že naše původní implementace je provedena s využitím knihovny Direct3D byla i tato knihovna zvažována pro provádění vzdáleného výpočtu. Knihovna Direct3D byla zvolena pro implementaci, neboť jako jediná bude plně podporována i budoucími operačními systémy Windows, jakožto hlavní operační systém na KIV FAV ZČU. V kombinaci s operačním systémem Microsoft Windows XP však bylo zjištěno, že:

- knihovna Direct3D neposkytuje akceleraci pomocí GPU, pokud je stroj spravován pomocí vzdálené plochy. Uvedená vlastnost znemožňuje použít dedikovaného stroje či strojů bez monitoru pro výpočet na GPU využívající Direct3D, neboť není možné kontrolovat stav výpočtu. Pokus o kontrolu tohoto stavu by okamžitě znamenal ztrátu veškerých dat.
- knihovna Direct3D je velmi citlivá na situace jako zamknutí počítače, spuštění šetřiče obrazovky či aktivaci systémového menu. Provedeni libovolné akce znamená ztrátu veškerých dat v paměti grafické karty a tedy i znehodnocení výpočtu. Ačkoliv je možné zamezit některým z těchto operací, např. spuštění šetřiče obrazovky, není možné uchránit běžný výpočetní stroj od vlivu uživatele.
- knihovna Direct3D není schopna poskytovat akceleraci, pokud je počítač uzamčen či není nikdo přihlášen do grafického prostředí. V praxi to znamená, že je velmi komplikované použít knihovnu Direct3D s běžným rozhraním pro distribuovaný výpočet, např. MPI, bez rozsáhlé manuální inicializace výpočetních uzlů.
- knihovna Direct3D není kompatibilní s operačním systémem Linux, což znemožňuje její využití s výpočetními prostředky dostupnými v rámci projektu METACentrum.

Na základě výše uvedených nedostatků knihovny Direct3D byla vyzkoušena i knihovna OpenGL. Pro praktické použití knihovny OpenGL na vzdáleném stroji je nutné, aby tento stroj měl spuštěnu instanci X-serveru s instalovanou akcelerací grafické karty a možností vytvářet a používat okna pro uživatele, pod kterým je spuštěn distribuovaný výpočet. Pokud je stroj takto připraven, je možné vzdáleně, pomocí konzole, spustit úlohu, provést výpočet a získat data. Pro ověření předpokladu byl proveden pokus, kdy bylo vzdáleně, pomocí SSH konzole, spuštěna úloha, která otevřela výstupní okno pomocí knihovny GLUT,

vykreslila jednoduchý obraz a vytvořil kopii části obrazu v operační paměti. Po spuštění byl získán předpokládaný obraz.

Lze tedy předpokládat, že bude možné provést danou operaci i pomocí prostředků pro podporu distribuovaného výpočtu, např. MPI. Zásadním problémem, který však brání použití OpenGL v distribuovaném výpočtu je absence grafických karet dostatečné funkcionality ve výpočetních prostředcích poskytovaných v rámci projektu METACentrum.

Kapitola 4

Závěr

V rámci řešení aktivity byl nalezen adekvátní způsob distribuce syntézy digitálních hologramů a byla vytvořena jeho efektivní implementace v prostředí MPI. Vzhledem k striktnímu dodržování ANSI C++ bylo dosaženo jisté úrovně platformní nezávislosti a je možné kód spouštět v operačních systémech Windows i Linux, který je v oblasti distribuovaných výpočtu často použit pro správu výpočetních uzlů.

Programové vybavení bylo úspěšně nasazeno na výpočetním bloku Hydra umístěného na ZČU spravovaného v rámci projektu METACentrum. V budoucnu se plánuje otestovat výpočet na větších výpočetních blocích, které jsou také v METACentru zapojeny.

Dalším stupněm bude agregace obou akceleračních přístupu, tj. distribuce a GPU. Toto řešení nám umožní začít počítat mnohem větší hologramy, než bylo doposud možné.

Literatura

[HJS07] I. Hanák, M. Janda, and V. Skala. Full-parallax hologram synthesis of triangular meshes using a graphical processing unit. In *3DTV Conference proc.*, pages ?–?, 2007.

[JHS06] M. Janda, I. Hanák, and V. Skala. Digital HPO hologram rendering pipeline. In *EG2006 short papers conf. proc.*, pages 81–84, 2006.

[JHS07] M. Janda, I. Hanák, and V. Skala. Hpo hologram synthesis for full-parallax reconstruction setup. In *3DTV Conference proc.*, pages ?–?, 2007.

[KZG07] P. Kaufmann, M-R. Ziegler, and M. Gross. A framework for holographic scene representation and image synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):403–415, 2007.

[Mic07] Microsoft directx 9.0 sdk (april 2007). WWW http://msdn2.microsoft.com/en-us/library/bb219740.aspx, 2007.

[SA06] M. Segal and K. Akeley. The opengl graphics system: A specification. WWW http://www.opengl.org/registry/doc/glspec21.20061201.pdf, 2006.

Dodatek A

Skripty

PBS skript pro spouštění úlohy v prostředí poskytovaném projektem METACentrum.

```
# zacatek
# PBS direktivy pro posilani maliu v pripade padu a ukonceni ulohy
PBS -m ae
PBS -M user123@kiv.zcu.cz
# vypise pridelene vypocetni uzly
cat $PBS_NODEFILE
#
APP=HoloFullSynthMPI
HOME=/home/user123/holography/task123
WORK=/scratch/user123/task123

WORKDATA=$WORK/Data
WORKOUT=$WORK/OUT
HOMELOG=$HOME/LOG
HOMEDATA=$HOME/Data
HOMEOUT=$HOME/OUT

# kopirovani vstupnich dat na vsechny vypocetni uzly
# spoustena binarka je tady s.mpi v jinem adresari
cd $HOME;
for host in `cat $PBS_NODEFILE` ; do
        rsh $host "mkdir $WORK";          #work root
        scp $APP $host:$WORK;             #application
      rsh $host "mkdir $WORKDATA";        #work data
      scp $HOMEDATA/* $host:$WORKDATA;
      rsh $host "mkdir $WORKOUT";         #output
done
# priprava konfiguracniho souboru pro MPI
#cp -Rpf * $WORK;
cd $WORK;
CPUS=`cat $PBS_NODEFILE | wc -l | awk '{print $1}'`
cat $PBS_NODEFILE > machines
# spousteni MPI jobu
time mpirun -machinefile ./machines -np $CPUS $APP > LOG 2>&1
# uklid -- presun vysledku
for host in `cat $PBS_NODEFILE` ; do
    scp -r $host:$WORK/*.log $HOMELOG;          #logs
    scp -r $host:$WORKOUT/* $HOMEOUT;           #results
    rsh $host "rm -Rf $WORK";             #total cleanup
done
# konec
```

Seznam obrázků

Seznam tabulek