PARALELNÉ VÝPOČTOVÉ PROSTREDIE DATAFLOW

(PARALLEL DATAFLOW COMPUTING ENVIRONMENT)

Martin GROFČÍK
Katedra počítačov a informatiky, Fakulta elektrotechniky a informatiky Technickej univerzity v Košiciach, Letná 9, 042 00 Košice, tel. 0905209141, E-mail: grofcik@hotmail.com

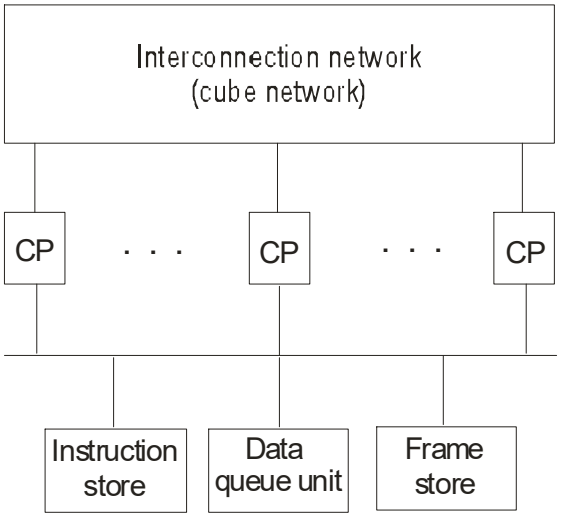
SUMMARY

The article deals with design and implementation of the parallel computational environment, which is driven by flow of the data. Design is based on the abstract model of the dataflow computer. This computer is developed at the Dapartment of computers and informatics, Faculty of Electrical engeneering and Informatics, Technical University of Košice (DF KPI)[1],[2]. Parallel computational environment presents design of the computational environment, which devlopment is based on the results of the simulation [3]. It modifies abstract design DF KPI to achieve unified workload on the components of the computer and to minimize count of the parallel access to components of the DF KPI. At the end of the document is presented implementation of the proposed design at the computer net environment, which is most achievable environment for parallel computation.

Keywords: *simulation, data flow, computer nets, parallel dataflow environment.*

1. ÚVOD

Prínosom článku je publikovanie architektúry výpočtového prostredia data flow. Architektúry počítačov data flow (DF) sú založené na výpočtovom modeli DF, v ktorom každá inštrukcia programu je pripravená na spustenie, pokiaľ na jej vstupe sú prístupné operandy. Počítače DF majú potenciál na využitie všetkých možností rozparalelnenia úlohy, ktoré existujú v programe. Program pre počítače DF je obvykle reprezentovaný grafom DF. Graf DF je vo všeobecnosti cyklický orientovaný graf, kde uzly sú operátory a hrany reprezentujú tok dát medzi produkujúcim a konzumujúcim uzlom. Pri spracovaní vstupných údajov (tokenov), sa dáta vyberajú zo vstupných front, odkiaľ sa mažú a výsledky zapisujú sa do výstupných frontov, ktoré môžu byť vstupom do iného operátora.



Obr. 1 Bloková schéma architektúry DF KPI
Fig. 1 Block scheme of the DF KPI architecture

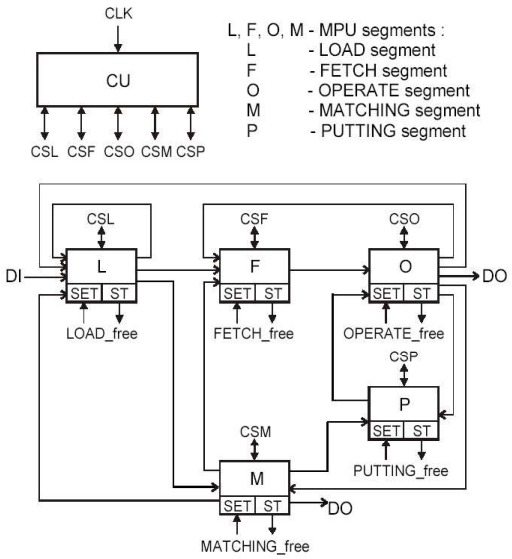
2. ARCHITEKTÚRA POČÍTAČA DATAFLOW KPI

Bloková schéma architektúry DF KPI je znázornená na Obr. 1. Jednotkou spracúvajúcou tokeny a produkujúcou tokeny na výstupe je koordinačný procesor (CP). Bližší popis CP je uvedený v podkapitole 2.1. Prepojovacia sieť (interconnection network (IN)) slúži na prenos tokenov medzi CP. Pamäť inštrukcií (instruction store (IS)) uchováva graf DF, ktorý sa používa na spájanie tokenov na vstupe operátora. Fronta údajov (data queue unit (DQU)) prepája výstupy z CP na ich vstupy, pričom dochádza k spájaniu tokenov do tzv. aktivačných rámcov. Tie sú následne vstupom do CP, kde je vzkonaný operátor grafu DF. Na ukladanie vytváraných aktivačných rámcov slúži pamäť rámcov (frame store (FS)).

2.1 Koordinačný procesor

Ako bolo spomenuté koordinačný procesor konzumuje na vstupe tokeny. V koordinačnom procesore sa následne inicializuje spájanie tokenov do aktivačných rámcov. Pokiaľ je aktivačný rámec vyplnený tokenmi, to znamená, že na vstupe sú všetky operandy, tak prebehne výpočet uzla grafu DF a na výstupe CP sa produkuje výstupný token. Bloková schéma procesora je znázornená na Obr. 2. Vstupom do procesora je segment *Load*, ktorý zo vstupu načítava token. Ďalej je token spracovávaný segmentom *Matching*, ktorý pomocou FS spája tokeny do aktivačných rámcov. Ak je aktivačný rámec vyplnený teda na vstupe operátora sú všetky operandy, spracovanie pokračuje segmentom *Fetch,* kde sa vyberie inštrukcia, ktorú reprezentuje uzol grafu DF. Uzol grafu DF spolu s údajmi v aktivačnom rámci je následne predaný segmentu *Operate*, kde prebehne skonzumovanie údajov a vyprodukujú sa výstupné údaje. Podľa potreby rozmnožovania údajov na výstupe, teda keď

výstupné dáta sú vstupom viacerých operátorov grafu DF, prebehne rozmnoženie údajov v segmente *Putting*.



Obr. 2 Bloková schéma koordinačného procesora
Fig. 2 Block scheme of the coordinating processor

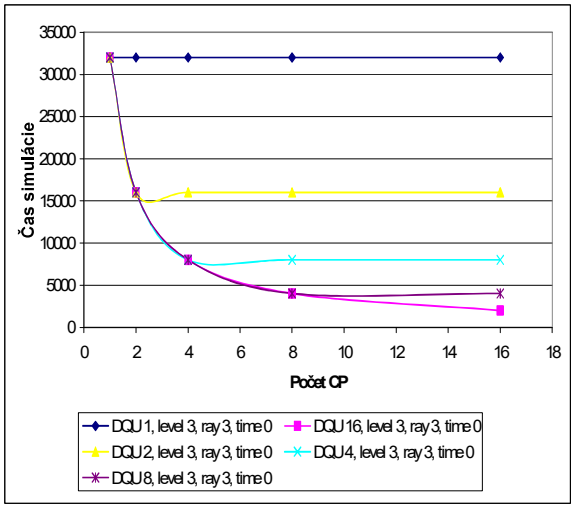
Na blokovej schéme je uvedený koordinačný procesor so spájaním aktivačných značiek na vstupe. V rámci výskumu architektúr DF na KPI boli navrhnuté aj iné varianty riešenia CP. Najjednoduchšou modifikáciou je presunutie segmentu *Matching* zo vstupu na výstup CP. Teda spájanie tokonov neprebieha pri vstupe tokenu do spravovania CP ale na výstupe, keď sa token zaraďuje na vstupnú hranu uzla grafu DF. Koordinačný procesor so spájaním operandov na vstupe slúži ako príklad pre prezentáciu štruktúry procesora schopného spracovať výpočet riadený tokom údajov. Ako je možné vidieť z prezentovaného návrhu, počet prepojení v CP medzi jednotlivými jeho segmentami je veľký to znamená že jeho riadenie bude komplikované. Požiadavka na počet paralelných prístupov k jednotlivým segmentom CP je závislá od paralelizmu, ktorý je možné dosiahnuť riešením úlohy. Pri možnostiach , ktoré poskytuje výpočet riadený tokom údajov sa dá predpokladať, že potreba paralelne komunikovať medzi segmentami bude vysoká. Uvedené fakty zneprístupňujú využitie prezentovanej architektúry kvôli cenovej nevýhodnosti.

3. VÝPOČTOVÉ PROSTREDIE DATAFLOW

Ako bolo spomenuté v predchádzajúcej kapitole, cenová nevýhodnosť riešenia zabraňuje rozširovaniu architektúry vo väčšom meradle. Pre minimalizáciu počtu prepojení jednotlivých segmentov CP a tým aj zjednodušenia riadenia CP je potrebné vykonať simulácie koordinačného procesora.

3.1 Výsledky simulačných experimentov

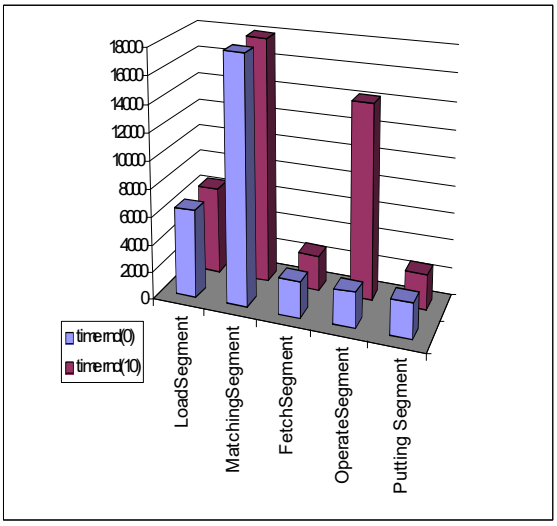
Na základe poznatkov, ktoré sa získali z vykonaných simulačných experimentov upraviť návrh CP. Model koordinačného procesora, na ktorom boli simulačné experimenty realizované bol vybratý model CP so spájaním operandov na vstupe. Program, ktorý bol na modeli počítača simulovaný, bola úloha fotorealistického zobrazovania 3D scén pomocou metódy sledovania lúča. Správanie sa počítača DF KPI pri zväčšovaní počtu CP zapojených do výpočtu je zobrazené na Obr. 3.



Obr. 3 Vplyv obmedzenia paralelného prístupu k DQU
Fig. 3 Restriction to connect DQU in parallel

Na uvedenom obrázku vidíme, že pokiaľ nie je obmedzený počet paralelných prístupov k DQU tak pridanie dvojnásobného počtu CP do výpočtu spôsobí približne dvojnásobné skrátenie času výpočtu. Samozrejme že pri výpočte musí byť yohľadnené naplnenie segmentov zreťazenia, preto urýchlenie výpočtu nie je dvojnásobné. Uvedený fakt platí len pri predpoklade že počet paralelných prístupov k jednotlivým komponentom DF KPI je neobmedzený, V príklade na obrázku sú zobrazené grafy popisujúce správanie systému pri obmedzení počtu paralelných prístupov k DQU. V grafe je počet paralelných prístupov k DQU označený v legende parametrom DQU (1 .. 16). Ako je možné vidieť z grafu obmedzenie paralelného prístupu k DQU na jeden CP, teda k DQU nemôžu pristupovať CP paralelne ale len sekvenčne spôsobí degradovanie schopnosti systému urýchľovať výpočet pridávaním CP do systému. Je zrejmé, že uvedená vlastnosť je závislá aj od času výpočtu v ostatných segmentoch CP. Umožnením paralelného prístupu viacerých CP k DQU je možné dosiahnuť urýchľovanie výpočtu ale opätovne len do tej miery, na koľko to umožnia paralelné prístupy a nie počet procesorov. K podobnému výsledku je možné dospieť, pokiaľ sa obmedzí počet paralelných prístupov k FS prípadne iným zdrojom systému DF KPI. Obmedzenie

paralelného prístupu k DQU najviac ovplyvňuje dĺžku trvania výpočtu úlohy. Pokiaľ predĺžime čas produkovania tokenov v segmente Putting skrátenie výpočtu pri pridaní dvojnásobného počtu počítačov to ovplyvní len v malej miere. Z uvedených výsledkov simulačných experimentov vyplýva, že počet paralelných prístupov vo veľkej miere ovplyvňuje vlastnosti systému čo sa týka možností rozparalelnenia úlohy a tým aj urýchlenia výpočtu. Obmedzenie počtu paralelných prístupov k jednotlivým častiam systému je dôležitým faktorom ovplyvňujúcim výslednú náročnosť riešenia a tým aj jeho prístupnosť. Minimalizáciu počtu paralelných prístupov ku komponentom systému je možné realizovať zapojením väčšieho počtu komponentov realizujúcich požadovanú funkčnosť tak, aby nevynikli neuspokojiteľné požiadavky na organizáciu riadenia. Spomínané fakty súvisia s možnosťami realizovania komunikácií v konkrétnej implementácii architektúry. Všeobecným predpokladom však je možnosť jednoduchého a robustného konfigurovania systému.

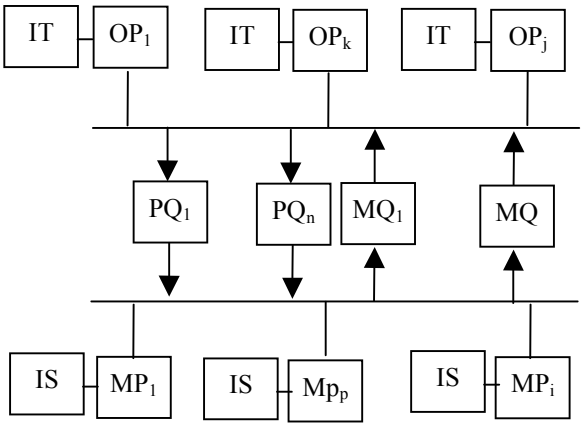


Obr. 4 Zaťaženie segmentov procesora
Fig. 4 Workload on segments of CP

V nasledujúcej časti bude popisovaná záťaž kladená na jednotlivé segmenty CP pri vykonávaní výpočtu. Záťaž kladená počas simulačného experimentu na segmenty CP je zobrazená na Obr. 4. Obrázok opisuje záťaž kladenú na jednotlivé segmenty CP pri výpočte fotorealistického zobrazenia 3D scény. Ako vyplynulo z ďalších experimentov, segmenty sú zaťažované rôzne. Miera ich zaťaženia je ovplyvnená výpočtovou náročnosťou jednotlivých uzlov grafu DF, a náročnosťou spájania operandov na vstupe do uzla grafu DF. Uvedené dve kritériá zaťažujú rôzne segmenty CP. Podľa uvedených kritérií je možné rozdeliť segmenty na operačné (zabezpečujúce spracovanie operátora v grafe DF) a spájacie (zabezpečujú spárovanie tokenov na vstupe do uzla grafu DF). Medzi operačné segmenty patrí segment *Operate*, *Fetch*. Spájacie segmenty sú *Matching*, *Load* a *Putting*. Na základe uvedených skutočností bolo možné vytvoriť návrh architektúry výpočtového prostredia DF.

3.2 Výpočtové prostredie dataflow

Bloková schéma výpočtového prostredia DF je znázornená na Obr. 5. Segmenty operačné sú reprezentované procesom operate (OP). Segmenty spájacie sú reprezentované procesom matching (MP). Okrem uvedených dvoch procesov v blokovej schéme na ich prepojenie slúži vkladací front (putting queue (PQ)) a spájací front (matching queue (MQ)). Na uchovanie grafu DF je určená jednotka pre uchovanie inštrukcií (instruction store (IS)). Jednotka uchovávajúca informácie o operátoroch v grafe DF je inštrukčná tabuľka (instruction table (IT)).



Obr. 5 Architektúra paralelného výpočtového prostredia DF
Fig. 5 Architecture of the parallel computational dataflow environment

Architektúra spracuje výpočet uzla grafu DF nasledovne: Vyplnený aktivačný rámec, obsahujúci aktivačné značky, vstupuje do OP. Tam na základe typu uzla grafu DF je vybratá inštrukcia, operátor ktorý spracuje vstupné dáta z aktivačného rámca. Na výstupe je produkovaný token, ktorý sa zapisuje do PQ. Na základe rozhodovacieho pravidla sa token z PQ zapíše do MP kde je zaradený do príslušných aktivačných rámcov. Príslušnosť tokenu na vstup do operátora grafu DF sa určuje na základe uzla grafu DF z ktorého token vystupuje a grafu DF uloženého v IS. Token je uložený do aktivačných rámcov prislúchajúcim inštrukciám na ktoré je smerovaný. Pokiaľ na vstup MP prichádza token ukončujúci vypĺňanie aktivačného rámca je aktivačný rámac predaný MQ, ktorá aktivačný rámec prenesie do OP, ktorý je schopný spracovania.

4. ZÁVER

Uvedená architektúra umožňuje znížiť počet paralelných prístupov k častiam počítača tak, že do výpočtu je možné zaradiť viac prvkov rovnakého typu a tým znížiť záťaž kladenú na jeden prvok počítača. Je zrejmé, že uvedený prístup nie je možné

aplikovať na neobmedzene veľké množiny komponentov počítača, pretože následkom zvyšovania počtu prvkov vznikajú zvýšené požiadavky na riadenie systému. Architektúra zjednodušuje priebeh výpočtu uzla grafu DF, od spájania až k spracovaniu aktivačného rámca výpočet prebieha bez návratov k už vykonaným segmentom. Teda výpočet prebieha v jednom prúde. V prípade možnosti zreťazenia výpočtu je uvedená skutočnosť jednou z podmienok pre nepretržité kontinuálne využitie všetkých stupňov zreťazenia. Oddelením operačných segmentov vznikla možnosť zapojiť do výpočtu problémovo orientované jednotky (napr. vektorové počítače) na výpočet špeciálnych operátorov v grafe DF (napr. násobenie matíc). Vyčlenenie spájacích segmentov umožňuje využitie dátového rozparalelnenia. Rozparalelnenie dátovej domény poskytuje možnosť zrátať úlohy veľkej pamäťovej náročnosti. V takom prípade do výpočtu sú zaradené viaceré segmenty pre spájanie operátorov.

Uvedenú architektúru je možné implementovať v rôznych prostrediach. Prostredím v súčastnosti najdostupnejším pre paralelné spracovanie úloh je počítačová sieť. Tá bola použitá pre implementáciu navrhnutej architektúry. Implementovanie v uvedenom prostredí je úzko späté s možnosťami implementovania komunikácie. Implementácia architektúry bola vykonaná prostredníctvom Common Object Request Broker Architecture (CORBA 0), pričom ako Object Request Broker (ORB) bol použitý Orbit 2 a operačný systém HP-UX11e. Popis rozhraní k jednotlivým procesom bol uskutočnený v Interface definition language (IDL). Navrhovaný systém umožňuje spájanie viacvstupových operátorov a zároveň definovanie vlastnej množiny operátorov čo umožní reguláciu jemnosti rozparalelnenia programátorom. Operátory sú v súčastnosti reprezentované funkciami v knižniciach v programovacom jazyku C, je však možné realizovať implementáciu v iných programovacích jazykoch a na ich základe definovať skupiny operátorov použitých v grafe.

Výpočtové prostredie DF umožňuje rozparalelnenie výpočtového procesu bez explicitného určenia výpočtového prostriedku, ktorý bude výpočet realizovať. Výpočtové prostredie je vhodné pre náročné výpočty, kde je možné rozdeliť vykonávanie DFG popisujúceho výpočet do podgrafov a operátorov a tak rozdeliť výpočet na menšie celky, ktoré medzi sebou komunikujú. Zároveň si výpočtové prostredie DF uchováva výhody DF systémov. Zníženie počtu paralelných prístupov k jednotlivým segmentom je závislé od konfigurácie.

LITERATÚRA

[1] Jelšina, M., Kollár, J.: The Dataflow Implementation Environment for Functional Languages, Proc. of Japan - Central Joint Workshop on Advanced Computing in Engineering, Pultusk (Poland), September 1994, 26 - 29.

[2] Jelšina, M.: The Dataflow Computer architecture with Direct Operand Matching, Journal of electrical Engineering, Vol. 46, 1995, 8, 279 – 285.

[3] Grofčík, M.: Simulation of the Raytracing problem on the Data Flow System, Proc. of ECI'02 Conference, FEI TU, Košice-Herľany 2002.

[4] Bolton, F.: Pure Corba, SAMS Publisher, ISBN 0672318121, jul 2001.

BIOGRAPHY

Martin Grofčík was born on 27.02.1976. In 1999 he graduated (MSc.) with honours at the department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. Is that time is PhD student at the same faculty. He performs his study at the external form since 2001. He was participating at the many projects of the concurrent computing. His scientific research is focusing on the parallel computers of the dataflow type and concurrent computing.