Žilinská univerzita v Žiline

Fakulta riadenia a informatiky

Martin Húdik, Ing.

Autoreferát dizertačnej práce

Modelovanie, optimalizácia a predikcia výkonnosti paralelných algoritmov

na získanie akademického titulu **„philosophiae doctor“ (v skratke PhD.)**

v študijnom programe doktorandského štúdia

aplikovaná informatika

v študijnom odbore:

9.2.9 aplikovaná informatika

Žilina, apríl 2013

Dizertačná práca bola vypracovaná v dennej forme doktorandského štúdia **na katedre technickej kybernetiky, Fakulte riadenia a informatiky Žilinskej univerzity v Žiline**

Predkladateľ: Ing. Martin Húdik
Žilinská univerzita v Žiline
Fakulta riadenia a informatiky
Katedra technickej kybernetiky

Školiteľ: prof. Ing. Ivan Hanuliak, CSc.
Žilinská univerzita v Žiline
Fakulta riadenia a informatiky
Katedra technickej kybernetiky

Oponenti: Doc. Ing. Ladislav Schwartz. PhD.
Žilinská univerzita v Žiline
Elektrotechnická fakulta
Katedra telekomunikácií a multimédií

prof. Ing. Imrich Rukovanský, CSc.
Vysoká škola logistiky o.p.s
Katedra logistiky a technických disciplín

Doc. Ing. František Huňka, CSc.
Ostravská univerzita v Ostravě
Přírodovědecká fakulta
Katedra informatiky a počítačů

Autoreferát bol rozoslaný dňa: .....

Obhajoba dizertačnej práce sa koná dňa ..... o ..... h. pred komisiou pre obhajobu dizertačnej práce schválenou odborovou komisiou v študijnom odbore **3.3.15 manažment, v študijnom programe manažment/ 9.2.9 aplikovaná informatika**, **v študijnom programe aplikovaná informatika**, vymenovanou dekanom Fakulty riadenia a informatiky Žilinskej univerzity v Žiline dňa .....

prof. Ing.Martin Klimo, PhD.
predseda odborovej komisie
študijného programu **aplikovaná informatika**
v študijnom odbore **9.2.9 aplikovaná informatika***
Fakulta riadenia a informatiky
Žilinská univerzita
Univerzitná 8215/1
010 26 Žilina

Úvod

Vysoká náročnosť skúmania a modelovania sústav matematiky, fyziky, biológie či chémie vo vede a technike alebo v komerčnej sfére multimediálne technológie, animácie a pod. , si vyžaduje neustále nové výpočtové prostriedky. Pre veľkú výpočtovú náročnosť takýchto úloh je výpočtový čas dlhý a nákladný. Práve nasadenie paralelných princípov sa ukazuje ako najefektívnejší spôsob riešenia. Využitie paralelných princípov otvára nové možnosti využitia výpočtovej sily počítačov. Používanie paralelných systémov prináša nové komplexné problémy, ktoré treba riešiť pre dosiahnutie požadovaného nárastu výkonnosti. Efektívne využitie paralelných počítačov sa dá dosiahnuť po splnení určitých predpokladov:

- dostupnosť vysoko-výkonných procesorov
- existencia vysoko rýchlostného prepojenia
- nástroje pre vývoj distribuovaných paralelných algoritmov
    - štandardizované vývojové prostredia – univerzálnosť paralelnej aplikácie
- existencia efektívneho paralelného algoritmu

Súčasný prechod k paralelným princípom je už dnes podporovaný technológiami (viacprocesorové, viacjadrové technológie, SMP) tiež aj programovým vybavením (MPI, OpenMP, PVM, Java, C#, Intel TBB). Rozvoju paralelných technológií napomáha rastúca priepustnosť prepojovacích sietí (Myrinet, Infiniband, Giga Ethernet, 10 Gigabit Ethernet, Fibre Channel) a takisto aj rozmach vysoko rýchlostných pripojení na internet (GRID).

Dominantným trendom súčasnosti je prechod od klasických masívnych paralelných počítačov k sieťam pracovných staníc (NOW). Dnešné viacjadrové osobné počítače a výkonné pracovné stanice zabezpečujú, že výkon takýchto sietí je dostatočný a ich cenová dostupnosť je oveľa prijateľnejšia ako u superpočítačov.

Paralelné princípy sú známe už od začiatku existencie počítačov, ale ich širšie využitie sa začalo až v poslednej dekáde. Teória sekvenčných algoritmov je dostatočne spracovaná a uspokojivo vyriešila niektoré problémy s nimi späté. Do pozornosti sa teraz dostáva teória paralelných algoritmov, ktorá je oveľa komplexnejšia z dôvodu väčšieho počtu parametrov, ktoré ovplyvňujú správanie a výkon paralelných algoritmov. Analýza výkonnosti paralelných systémov je oveľa zložitejšia ako optimalizácia celkovej doby vykonania pri sekvenčných algoritmoch. Potrebné je zohľadniť viacero faktorov ovplyvňujúcich výkon paralelných algoritmov ako sú :

- počet procesorov, výpočtových uzlov
- architektúra počítača
- dostupná pamäť
- priepustnosť komunikačných sietí
- veľkosť úlohy (dáta)
- a iné

Moja práca vychádza z analýzy paralelných princípov a je smerovaná do oblasti modelovania výkonnosti paralelných algoritmov na dostupných paralelných sieťach počítačov (NOW,SMP,GRID). Cieľom práce je prispieť k metodike vývoja efektívnych paralelných algoritmov, zložitosti paralelných algoritmov a spôsobu hodnotenia ich výkonnosti na paralelných sieťach počítačov.

Ciele dizertačnej práce

Analýza stavu u nás a vo svete

- paralelné počítače (PP)
- paralelný algoritmus (PA)
  - spoločná pamäť
  - distribuovaná pamäť
- paralelné vývojové prostredia
  - OpenMP, Java
  - MPI, Java
- sústava lineárnych rovníc (SLR)
  - sekvenčné algoritmy (SASLR)
  - paralelné algoritmy(PASLR)

Teoretická časť

- analýza SASLR
  - kritéria výberu
  - stabilita riešenia
  - výber SASLR
  - dekompozičné modely PASLR
- modelovanie zložitosti paralelných algoritmov (PA)
  - výpočtová
  - režijná (oneskorenia)
    - ✓ dekompozícia (paralelizácia)
    - ✓ komunikačná
    - ✓ synchronizácia
- optimalizácia PA
  - efektívne PA
- predikcia zložitosti

Experimentálna časť

- vývoj efektívnych PA
  - spoločná pamäť (SMP)
  - distribuovaná pamäť (NOW, Grid)
- implementácie, overenia
- experimentálne merania
  - SMP architektúry
  - NOW
  - Grid
- porovnanie nameraných
- štatistické vyhodnotenie

Zovšeobecnenia, závery, odporúčania

Prínosy

- teoretické
- praktické

1 Analýza paralelných počítačov

Počítačoví architekti sa vždy usilovali o zvýšenie výkonnosti počítačových architektúr. Vysoký výkon môže pochádzať z rýchlych integrovaných obvodov s veľkou hustotou integrácie alebo z použitia princípov paralelizmu. Jednoprocesorové super počítače dosiahli vysoké rýchlosti a dotlačili hardvér na technologický fyzický limit výroby čipov. Tento trend skončí, pretože limitom

sú fyzické hranice, ktoré obmedzujú výpočtový výkon jednoprocesorového systému. Táto práca sa bude venovať moderným počítačovým architektúram, ktoré využívajú paralelné prostredie viacerých výpočtových uzlov (procesory, jadrá, počítače).

1.1 Paralelné počítače

Klasické paralelné počítače, ktoré dominovali vo svete do roku 1990, sa na Slovensku nerozšírili a podľa posledných vývojových trendov vo svete sa už u nás ani nerozšíria. Vývojové trendy vo svete už niekoľko rokov smerujú jednoznačne k nahradzovaniu klasických superpočítačov skupinou vzájomne prepojených, úzko špecializovaných počítačov (klastrov) alebo výkonných pracovných staníc (NOW). Dôvod rozširovania klastrov vo svete je najmä ich univerzálnosť, vzájomná nezávislosť, cena a rozšíriteľnosť podľa aktuálnych požiadavok užívateľov. Nevýhodou je komplikovanejšie riadenie pre absenciu spoločnej pamäte. Paralelné systémy dosahujú vyšší výkon, ale ich vysoká cena bráni väčšiemu rozšíreniu.

Trendom v posledných rokoch boli viacprocesorové symetrické systémy, pričom počet použitých procesorov pre architektúru typu SMP stále narastal. Aktuálne začínajú dominovať viacjadrové (multi-core) procesory ako aj viacbunkové (cell) procesory. Príkladom je procesor TeraScale, ktorý dosahuje výkony v rádoch v teraflopov a počet jadier v budúcnosti bude rádovo v stovkách. Hlavnou výhodou viacjadrových procesorov je, okrem nárastu výkonnosti, podstatné zníženie spotreby procesora a teda ekonomickejšia prevádzka počítača.

Technológia Grid-ov sa v poslednom období zaraďuje medzi nastupujúce alternatívy distribuovaných paralelných počítačov. Predstavuje ľahko rozšíriteľnú architektúru, ktorá môže zahŕňať prakticky akékoľvek zariadenie schopné komunikácie.

Prehľad vývoja paralelných architektúr na báze osobných procesorov firmy Intel (NOW, SMP, Grid), možno zosumarizovať takto:

- jednoprocesorové (postupne s implementáciou paralelných princípov)
- viacprocesorové (multiprocesory - spoločná pamäť, distribuovaná pamäť)
- jednoprocesorové s paralelnými princípmi
- skalárne : od roku 1989 - Intel 486, superskalárne od roku 1993 Pentium
  - SIMD inštrukcie SSE (Streaming SIMD Extension) - od roku 1999 Pentium III
  - inovácia ISSE2 (Internet SSE) – od roku 2000 Pentium IV
  - deep pipelining (20 stupňové zreťazenie) – od roku 2000 Pentium IV
- viacprocesorové typu SMP (symetrické) v jednej pracovnej stanici
  - technická podpora už od roku 1978 - Intel 8086
  - podstatné prepracovanie v roku 1995 - Pentium Pro (v súčasnosti systémy Xeon)
  - EPIC (Explicit Parallel Instruction) – od roku 2001 (Xeon systémy)
  - technológia Multi-threading (logické znásobenie počtu jadier)
  - dvojjadrové (dual core) od roku 2005
  - štvorjadrové (quad core) od roku 2006
  - osemjadrové od roku 2010
  - desaťjadrové 2011
  - viacjadrové (50-80 jadier na jednom čipe), viacbunkové (multi-core, multi-cell)
  - Intel Xeon Phi coprocessor - 60 a viac jadier, každé jadro 4 thread
- viac počítačové – prakticky od nástupu osobných počítačov
  - lokálne siete osobných počítačov (PC LAN, cluster)
  - siete pracovných staníc NOW (Network of workstations)

2 Vývoj paralelného algoritmu

Konkrétna realizácia paralelného algoritmu alebo programu je podriadená výberu programovacieho modelu. Vo všeobecnosti časť tvorby paralelných algoritmov, ktoré nie sú závislé od špecifického programovacieho modelu, zahrňujú nasledovné činnosti:

- rozdelenie úlohy na jednotlivé nezávislé procesy (Decomposition - dekompozícia),
- priradenie jednotlivých úloh procesom alebo vláknam,
- namapovať procesy alebo vlákna jednotlivým fyzickým procesorom alebo jadrám resp. uzlom počítačovej siete na výpočet (Mapping - mapovanie),
- komunikácia medzi jednotlivými procesmi cez prepojenú počítačovú sieť IPC (Interprocess communications),
- riadenie prístupu ku zdieľaným dátam,
- synchronizácia procesov pri rôznych fázach výpočtu,
- optimalizácia výkonnosti - vyladenie (Tunning)[19,28,43,86].

3 Model nákladov v počítačoch s distribuovanou pamäťou

Celkový čas potrebný na prenesenie správy sa skladá z nasledujúci parciálnych časov:

- t_s – Štartovací čas – čas potrebný na spracovanie/prípravu správy na prijímacej aj odosielacej strane
- T_h – Per-hop čas – čas potrebný na prenesenie správy medzi dvoma susednými uzlami
- T_w – Per-word prenosový čas – ak šírka prenosového kanálu je *r* slov za sekundu, tak čas na prenos jedného slova je prevrátená hodnota $t_w = 1/r$

$$t_{comm} = t_s + l\,t_h + t_w m \qquad (3\text{-}1)$$

Táto rovnica predstavuje nákladový model pre komunikáciu pomocou správ o veľkosti *m* medzi uzlami systému vzdialených *l* skokov. Keďže vo väčšine systémov počet skokov *l* v prepojovacej sieti je malý, môžeme $l t_h$ (per-hop) čas zanedbať. Výsledný čas komunikácie „bod-bod“ je potom

$$t_{comm} = t_s + t_w m \qquad (3\text{-}2)$$

3.1.1 Model nákladov v počítačoch so spoločnou pamäťou

Odhad komunikačných nákladov pri počítačoch so spoločnou pamäťou je oveľa zložitejší. Do úvahy je treba zobrať veľa premenných na vyprodukovanie komplexného komunikačného modelu. Takýto model by bol veľmi špecifický a závislý na konkrétnej architektúre počítača a nebol by všeobecne použiteľný.

V zjednodušenom modeli sa všetky oneskorenia spojené s pamäťovými operáciami, oneskorenia spojné so sieťou a ostané oneskorenia pripočítajú do t_s konštanty. Táto t_s konštanta je spojená s úvodným prístupom k spojitému celku spoločných dát o veľkosti *m* slov. Ďalej predpokladáme, že prístup k spoločným dátam je časovo náročnejší ako prístup k lokálnym dátam a teda čas prístupu k jednému slovu t_w priradíme práve spoločným dátam. Z toho vyplýva, že pre zjednodušený model môžeme napísať rovnicu:

$$t_{comm} = t_s + t_w m \qquad (3\text{-}3)$$

Táto rovnica je totožná s rovnicou popisujúcou zjednodušený nákladový model pre počítače komunikujúce pomocou správ (3-2). Konštanty t_s a t_w sú pre model popisujúci počítače so spoločnou pamäťou oveľa menšie.

4 Hodnotenie výkonnosti

4.1.1 Klasické

4.1.1.1 Cena/výkon

Nech *T(s, p)* reprezentuje dobu riešenia paralelného algoritmu *p* procesormi (*s* definuje veľkosť daného problému). Potom cena tohto paralelného algoritmu sa môže definovať ako

$$C(s) = p\,T(s,p) \qquad (4\text{-}1)$$

C(s) reprezentuje celkovú prácu vykonanú všetkými procesormi zapojenými vo výpočte. Paralelný program nazývame cenovo optimálny vtedy keď *C(s) = T(s,1)*, t.j. keď celkovo vykoná rovnaké množstvo operácií ako najrýchlejší sekrečný algoritmus, ktorého čas výpočtu je *T(s,1)*.

4.1.2 Špecializované

4.1.2.1 Paralelné zrýchlenie

Nech *O(s,p)* je celkový počet jednotkových operácií, ktoré vykonáva p-procesorový systém pre aplikačnú úlohu s veľkosťou problému *s* a *T(s, p)* je doba paralelného vykonania úlohy. Vo všeobecnosti, *T(s, p) < O(s, p)* ak sa vykonáva viac ako jedna operácia *p* procesormi za časovú jednotku, pričom *p ≥2*. Predpokladajme *T(s, 1) = O(s, 1)* pre jednoprocesorový systém (klasický sekvenčný systém). Paralelné zrýchlenie (speedup) je potom definované ako

$$S(s,p) = \frac{T(s,1)}{T(s,p)} \qquad (4\text{-}2)$$

4.1.2.2 Efektívnosť

Efektívnosť systému pre p - procesorový systém je definovaná ako

$$E(s,p) = \frac{S(s,p)}{p} = \frac{T(s,1)}{p\,T(s,p)} \qquad (4\text{-}3)$$

Hodnota *E(s, p)*, ktorá sa pre určité *p* rovná približne *1* signalizuje, že takýto paralelný systém, ktorý používa *p* procesorov, sa vykonáva približne *p* krát rýchlejšie v porovnaní so sekvenčným algoritmom. Platí, že *E(s, p*) je v intervale $0 < E(s,p) \leq 1$, pričom ak *E = 1* zodpovedá teoreticky najlepšiemu zrýchleniu výpočtu, ak *S (s, p) = p*.

4.1.2.3 Izoefektívnosť

Pracovná záťaž w algoritmu sa často zvyšuje podľa rádu *O(s)*, kde *s* je vstupná veľkosť problému. Z toho dôvodu definujeme pracovnú záťaž *w = w(s)* ako funkciu *s*. V paralelných výpočtoch je veľmi užitočné definovať funkciu izoefektívnosti ako závislosť vstupnej pracovnej záťaže od mohutnosti paralelného počítača, ktorá je nevyhnutná pre dosiahnutie konštantnej efektívnosti *E* pri implementácii paralelného algoritmu na danom paralelnom počítači. Nech *h* sú celkové režijné oneskorenia (architektúra, riadenie, komunikácia), ktoré sú spojené s implementáciou daného algoritmu na konkrétnom paralelnom počítači. Tieto oneskorenia sú funkciou tak veľkosti paralelného počítača, ako aj veľkosti problému, t. j. *h = h (s, p)*. Efektívnosť paralelného algoritmu, ktorý je implementovaný na danom paralelnom počítači je potom definovaná ako

$$E(s,p) = \frac{w(s)}{w(s) + h(s,p)} \qquad (4\text{-}4)$$

Pracovná záťaž *w(s)* odpovedá užitočným výpočtom, zatiaľ čo oneskorenie *h(s, p)* tvoria režijné časy (riadenie, synchronizácia, komunikáciaTeoretická časť práce

5 Paralelné algoritmy

5.1 Spoločná pamäť

Pri paralelných algoritmoch oproti sekvenčným algoritmom je nutné pri paralelnom prístupe k spoločným dátam v pamäti prijať riadiace mechanizmy, ktoré vyvolávajú prídavné oneskorenia. Pri modelovaní výkonnosti paralelných algoritmov pre klasické paralelné systémy so spoločnou pamäťou sa tieto režijne náklady zanedbávajú pretože sa predpokladá, že sú v porovnaní s náročnosťou vykonávaného výpočtu podstatne nižšie. Preto pre paralelné algoritmy so spoločnou pamäťou je časová zložitosť spravidla časovou zložitosťou výpočtu podobne ako pri sekvenčnom algoritme. Premietnutie tohto predpokladu do vzťahu pre asymptotickú funkciu izoefektívnosti znamená:

$$\omega(s) = \max [\text{výpočet}, h(s,p) < \text{výpočet}] = O(\text{výpočet}) \qquad (5\text{-}1)$$

- w –pracovná záťaž
- s - vstupná veľkosť problému
- h – oneskorenia

5.2 Distribuovaná pamäť

Distribuované paralelné algoritmy (DPA) pre súčasné paralelné počítače na báze NOW, SMP, Grid vyžadujú pre modelovanie výkonnosti komplexnú analýzu rozsahu vplyvov všetkých podstatných komponentov na modelovanie výkonnosti, a to

- vplyv architektúry paralelného systému
- vplyv medziprocesorovej komunikácie IPC (Interprocess communication)
  - inicializácia komunikácie (start-up time)
  - vlastný prenos dát
  - smerovanie (prenos cez viac komunikačných uzlov)
- vlastný výpočet.

V dôsledku týchto vplyvov je analýza celkového oneskorenia paralelného výpočtu

$$T(s,p) = T_{comp} + T_{par} + T_{interact}, \qquad (5\text{-}2)$$

, kde T_{comp}, T_{par}, $T_{interact}$ udávajú jednotlivé oneskorenia pre vykonanie výpočtu, réžiu pre paralelizáciu a komunikáciu procesov. Časti T_{par}, $T_{interact}$ vytvárajú vo vzťahu pre izoefektívnosť funkciu režijných a komunikačných oneskorení *h(s, p)*, ktorej vplyv pre distribuované paralelné algoritmy (DPA) nie je možné zanedbať, pretože môže byť pre celkovú zložitosť algoritmu dominantná.

Pre analýzu asymptotickej izoefektívnosti distribuovaných paralelných algoritmov (DPA) všeobecne platí

$$\omega(s) = \max [\text{výpočet}, h(s,p)] \qquad (5\text{-}3)$$

5.3 Medziprocesná komunikácia

Všetky paralelné algoritmy vyžadujú medziprocesnú komunikáciu. Vplyv tejto komunikácie vo veľkej miere ovplyvňuje výkonnosť paralelného algoritmu. Pre optimálny návrh a tak isto aj pre analyzovanie paralelného algoritmu, je potrebné poznať komunikačné mechanizmy a ich odhad časovej náročnosti.

Pri analyzovaní jednotlivých typov komunikácie vychádzame z komunikácie „bod-bod“, pre ktorú, ako bolo uvedené v kapitole 3, je časový prenos $T = t_s + t_w m$.

V reálnych podmienkach komunikácia prebieha medzi viacerými uzlami siete. Tieto typy prepojení uzlov v sieti sa v praxi nenachádzajú často, ale práve niektoré algoritmy komunikujú takýmto spôsobom, pričom takáto komunikácia je efektívnejšia pre možnosť serializácie (pipelining) komunikácie.

- Linear array – práca s riadkami alebo stĺpcami matice
- Mesh – Rozšírenie lineárneho zoznamu do druhého rozmeru
  Hypercube[1] – veľa algoritmov pracujúcich s rekurziou sa prirodzene mapujú na hypercube.

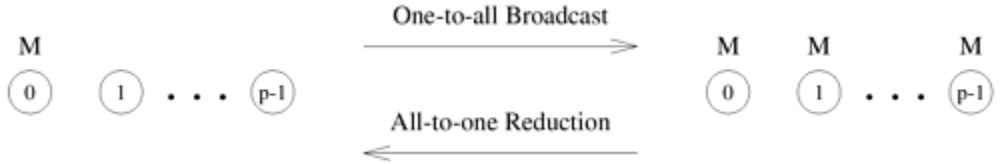
Východisko $t_s + t_w m$ platí pre väčšinu paralelných architektúr počítačov s tým, že pre každú architektúru sú rôzne konštanty. V určitých prípadoch však nastáva zahltenie komunikačnej siete a spomínaná rovnica už neplatí.

5.3.1 One-To-All(broadcast) a All-To-One redukcia

Paralelný algoritmus často potrebuje zaslať identické dáta všetkým procesom alebo určitej podmnožine procesov. Takáto operácia sa nazýva one-to-all vysielanie (broadcast).

Pri one-to-all vysielaní iba jeden proces má dáta o veľkosti *m*, ktoré budú rozoslané. Na konci operácie bude existovať *p* kópii pôvodných dát, ktoré boli rozoslané každému procesu. Opačná operácia sa nazýva all-to-one redukcia. Pri all-to-one redukcií každý *p* prispievajúci proces obsahuje zásobník M obsahujúci dáta o veľkosti *m*. Dáta zo všetkých procesov sa skombinujú cez asociatívnu operáciu a následne sú kumulované v cieľovom procese. Táto komunikačná metóda slúži napríklad na nájdenie maxima, minima, sumy atď.

Tieto metódy sa používajú v mnohých paralelných algoritmoch, napr. pri maticovom a vektorovom násobení, Gaussovej eliminácií, nájdení najkratšej cesty a pod.



Obr.5-1 All-to-one vysielanie a One-to-all redukcia

Algoritmus:

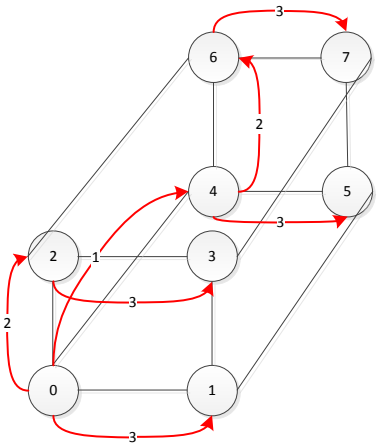
One-to-all, ako neefektívna implementácia tejto komunikačnej operácie by bolo rozposielať správy z jedného procesu do zvyšných *p-1* procesov sekvenčne „bod-bod“. V jednom časovom okamihu by správa došla iba jednému procesu. Pri použití spoločnej zbernice sa používa práve tento postup. Riešením tohto problému je použitie metódy rekurzívneho zdvojenia. Pri rekurzív-

[1]Každý uzol je prepojený s n ďalšími a geometricky znázornené, tvorí teleso v euklidovskom priestore so štyrmi a viac dimenziami, zodpovedajúce kocke v trojrozmernom priestore

nom zdvojení štartovací proces odošle správu jednému procesu a ten nasledovne odošle správu tiež ďalšiemu procesu. V každom kroku sa počet odosielateľov správy zdvojnásobuje. Na dokončenie celého vysielania je potrebné $log_2 p$ krokov.

Hypercube

Hypercube si predstavujeme ako rozšírenú mesh sieť do *d* dimenzií. Algoritmy používané pre mesh sa dajú aplikovať a to postupne pre každú dimenziu hypercubu. Komunikácia sa začína v uzle 0.



Obr.5-2 One-to-all komunikácia na hyperkocke

Na rozdiel od lineárneho zoznamu pri hypercube nezáleží na poradí komunikácie. Výsledná časová náročnosť operácie bude vždy rovnaká, nedochádza k zahlteniu siete.

5.3.2 Vyhodnotenie

V tabuľke Tab. 5-1 je uvedený prehľad komunikačných operácií a ich časová náročnosť pre komunikačnú štruktúru hypercube. Hypercube predstavuje jeden z najlepších možných komunikačných prepojení, ktorý však v praxi nie je často realizovaný. Pri budovaní finančne menej náročných paralelných systémov typu NOW sa najčastejšie používa interconnected prepojenie (vzájomné prepojenie lineárny zoznam).

Tab. 5-1Sumár komunikačných časov pre jednotlivé operácie

Operácia	Čas
One-to-all vysielanie, All-to-one redukcia	$\min((t_s + t_w m) \log_2 p, 2(t_s \log_2 p + t_w m))$
All-to-all vysielanie, All-to-all redukcia	$t_s \log_2 p + t_w m(p - 1)$
All-reduce	$\min((t_s + t_w m) \log_2 p, 2(t_s \log_2 p + t_w m))$
Scatter, Gather	$t_s \log_2 p + t_w m(p - 1)$
All-to-all personalized	$(t_s + t_w m)(p - 1)$

6 Sústava lineárnych rovníc

Majme sústavu lineárnych rovníc[4,45] (SLR) v tvare:

$$
\begin{aligned}
a_{0,0}x_0 + a_{0,1}x_1 + \cdots + a_{0,n-1}x_{n-1} &= b_{0,} \\
a_{1,0}x_0 + a_{1,1}x_1 + \cdots + a_{1,n-1}x_{n-1} &= b_{1,} \\
&\vdots \\
a_{n-1,0}x_0 + a_{n-1,1}x_1 + \cdots + a_{n-1,n-1}x_{n-1} &= b_{n.}
\end{aligned}
\qquad (6\text{-}1)
$$

V maticovej notácií môžeme danú sústavu zapísať ako

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b} \qquad (6\text{-}2)$$

Kde **A** je matica koeficientov, **b** je vektor pravých strán a **x** je vektor neznámych.

Riešiť sústavu znamená nájsť všetky riešenia tejto sústavy. Riešením sústavy (6-1) rozumieme vektor $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)^T$ pre ktorý platí $\boldsymbol{A} \times \boldsymbol{r} = \boldsymbol{b}$.

Majme sústavu lineárnych rovníc v tvare

$$\begin{pmatrix} a_{0,0} & a_{0,1} & \ldots & a_{0,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m-1,0} & a_{m-1,1} & \ldots & a_{m-1,n-1} \end{pmatrix} \times \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ \vdots \\ b_{n-1} \end{pmatrix}$$

ak platí m>n hovoríme, že sústavu rovníc je preurčená. Ak platí m<n hovoríme, že sústava rovníc je nedourčená. Ak vektor **b** je nulovým vektorom, tak sústavu označujeme ako homogénnu a má minimálne jedno riešenie $\boldsymbol{r} = (0,0, \ldots,0)^T$, ktoré nazývame triviálnym riešením sústavy.

Pri riešení sústavy lineárnych rovníc môže nastať jedna z týchto možností:

- sústava má iba jedno riešenie,
- sústava má nekonečne veľa riešení,
- sústava nemá žiadne riešenie.

Základný teorém zaoberajúci sa problémom riešiteľnosti sústavy lineárnych rovníc je Frobeniova veta, ktorá hovorí, že sústava má riešenie práve vtedy ak hodnosť matice sústavy je rovná hodnosti rozšírenej matice sústavy: $h(\boldsymbol{A}) = \mathrm{h}(\boldsymbol{A}|\boldsymbol{b})$

6.1 Vybrané algoritmy na výpočet SLR

Na výpočet sústavy lineárnych rovníc poznáme viacero metód, ktoré delíme na

- priame metódy (finitné)
  - Cramerovo pravidlo
- eliminačné metódy
  - Gaussova eliminácia
  - Gauss-Jordanová eliminácia
  - Iné (LU rozklad)
- a iteračné metódy.

6.1.1 Priame metódy (finitné)

Prieme metódy po konečnom počte krokov dospejú k riešeniu. Presnosť tohto riešenia môže byť ovplyvnená binárnou reprezentáciu reálnych čísel pri číslicovom spracovaní počítačom. Existujú metódy na odstránenie týchto numerických problémov, aby bola zachovaná numerická stabilita

Vo väčšine prípadov je systém lineárnych rovníc riešený v dvoch elementárnych krokoch. Prvý krok spočíva v algebrickej manipulácií s maticou, tak aby sa pôvodný systém lineárnych rovníc upravil na určitú formu trojuholníkovej matice. V druhom kroku otočíme smer práce s maticou

a vykonávame tzv. spätnú substitúciu. Po vykonaní druhého kroku dostávame riešenie sústavy (6-1) .

6.1.1.1 Cramerovo pravidlo

Vytvorme matice $\mathbf{A_1}$ až $\mathbf{A_n}$, n=1,2,3,... tak, že v pôvodnej matici **A** postupne nahradíme n-tý stĺpec matice vektorom **b**. Determinanty týchto matíc označme D_1 až D_n a determinant pôvodnej matice označme D. Riešenie sústavy dostávame nasledovne:

$$r_1 = \frac{D_1}{D},\ r_2 = \frac{D_2}{D},\ \dots\ , r_n = \frac{D_n}{D}, \qquad (6\text{-}3)$$

Cramerovo pravidlo vyžaduje výpočet (n+1) determinantov pre výpočet n x n sústavy lineárnych rovníc. Pre každý determinant n! násobení a n! sčítaní. Výpočtová zložitosť celého algoritmu je približne 2(n+1)!. Tato zložitosť je veľká (Tabuľka **Chyba! Nenalezen zdroj odkazů.**) a preto sa Cramerovo pravidlo používa iba na výučbové účely.

6.1.2 Eliminačné metódy

6.1.2.1 Gaussova eliminačná metóda (GEM)

Gaussova eliminačná metóda vychádza z predpokladu, že pomocou ekvivalentných úprav transformujeme **Ax = b** na ekvivalentnú **Cx = d** sústavu. **Ax = b** a **Cx = d** považujeme za ekvivalentné, ak každé riešenie sústavy **Ax = b** je riešením sústavy **Cx = d** a naopak. Transformovaná **Cx = d** sústava má nasledovnú vlastnosť

$$c_{i,j} = 0\ pre\ i > j, c_{i,i} \neq 0, \qquad (6\text{-}4)$$

Z tejto vlastnosti vyplýva, že matica C je regulárna horná trojuholníková matica. Spätnou substitúciou túto sústavu vyriešime a dostaneme tak riešenie pôvodnej sústavy **Ax = b**.

Ekvivalentné úpravy, používame pri GEM:

- vynásobenie niektorej rovnice sústavy (t. j. jej pravej i ľavej strany) nenulovým číslom
- pripočítanie násobku jednej rovnice sústavy k inej rovnici sústavy

6.1.2.2 Gauss-Jordanova eliminácia

Ide o variáciu GEM. Pri tejto metóde pracujeme s rozšírenou maticou [**A|b**]. Cieľom je upraviť túto rozšírenú maticu do tvaru diagonálnej matice. Postupne upravujeme prvky pod aj nad diagonálou matice, tak aby sme dostali výslednú jednotkovú maticu **I**. Pôvodný vektor **b** je po úprave rozšírenej matice na jednotkovú maticu transformovaný na vektor riešení **r**.

$$\left(\begin{array}{cccc|c} a_{0,0} & a_{0,1} & \dots & a_{0,n-1} & b_0 \\ a_{1,0} & a_{1,1} & \dots & a_{1,n-1} & b_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m-1,0} & a_{m-1,1} & \dots & a_{m-1,n-1} & b_{n-1} \end{array}\right) = \left(\begin{array}{cccc|c} 1 & 0 & \dots & 0 & r_0 \\ 0 & 1 & \dots & 0 & r_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & r_{n-1} \end{array}\right)$$

6.1.2.3 Eliminácia pomocou inverznej matice

Majme sústavu rovníc zapísanú v maticovom tvare (6-2), ak vynásobíme obe strany inverznou maticou $\mathbf{A^{-1}}$ dostávame nasledujúce rovnice:

$$\boldsymbol{A^{-1}Ax} = \boldsymbol{A^{-1}b}$$
$$\boldsymbol{Ix} = \boldsymbol{A^{-1}b};\ \boldsymbol{A^{-1}A} = \boldsymbol{I}$$
$$\boldsymbol{x} = \boldsymbol{A^{-1}b};\boldsymbol{Ix} = \boldsymbol{x}$$

Na výpočet sústavy lineárnych rovníc ***Ax=b*** určíme $\mathbf{A^{-1}}$ inverznú maticu a vynásobíme ju vektorom pravých strán ***b***. Ak matica ***A*** má inverznú maticu $\boldsymbol{A^{-1}}$ tak $\boldsymbol{A} \times \boldsymbol{A^{-1}} = \boldsymbol{I} = \boldsymbol{A^{-1}} \times \boldsymbol{A}$ a teda na urče-

nie inverznej matice musíme vyriešiť maticovú rovnicu $\boldsymbol{A^{-1} \times A = I}$. Na samotný výpočet môžeme použiť buď GEM alebo Gauss-Jordanovú eliminačnú metódu.

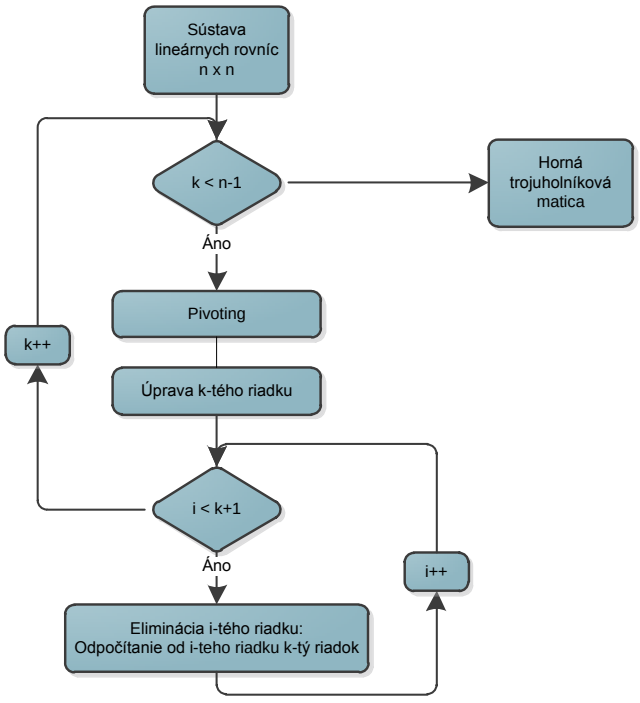
6.1.2.4 Porovnanie výpočtovej náročnosti jednotlivých metód

V tabuľke Tab.6-1 je zosumarizovaná výpočtová náročnosť jednotlivých metód na výpočet $n \times n$ sústav lineárnych rovníc.

Tab.6-1 Porovnanie výpočtovej náročnosti jednotlivých metód

Metóda	Násobenie	Sčítavanie
Gaussova eliminácia	$\frac{1}{3}n^3 + n^2 - \frac{1}{3}n$	$\frac{1}{3}n^3 + \frac{1}{2}n^2 - \frac{5}{6}n$
Gauss-Jordanova eliminácia	$\frac{1}{3}n^3 + n^2 - \frac{5}{2}n + 2$	$\frac{1}{3}n^3 - \frac{3}{2}n + 1$
Riešenie lineárnych rovníc za pomoci inverznej matice – GEM	$\frac{5}{6}n^3 + 2n^2 - \frac{5}{6}n$	$\frac{4}{3}n^3 - \frac{1}{2}n^2 - \frac{5}{6}n$
Riešenie lineárnych rovníc za pomoci inverznej matice – Gauss-Jordanova eliminácia	$\frac{3}{2}n^3 + \frac{1}{2}n - 1$	$\frac{3}{2}n^3 + \frac{1}{2}n^2 - \frac{1}{2}n$
Cramerovo pravidlo	$(n+1)!$	$(n+1)!$

V prípade ak n je malé tak porovnávanie výkonnosti jednotlivých metód je otázne, ale v prípade veľkých n lineárnych sústav nastáva situácia, keď sú rozdiely vo výpočtovej náročnosti signifikatné.



Obr.6-1 Vývojový diagram GEM (sekvenčný)

6.1.3 Sekvenčný algoritmus GEM

Cieľom tohto algoritmu je transformovať maticu **A** pomocou ekvivalentných úprav na hornú trojuholníkovú maticu.

Sekvenčná zložitosť

Gaussová eliminácia približne vykonáva $\frac{n^2}{2}$ delení a približne $\frac{n^3}{3} - \frac{n^2}{2}$ sčítaní a násobení. Pre zjednodušenie predpokladáme, že všetky atomické aritmetické operácie sú vykonávane v jednotkovom čase. Pri tomto predpoklade môžeme výpočtovú zložitosť pre veľké n napísať v tvare:

$$t_{sekv} = O\left(\frac{2}{3}n^3\right) \qquad (6\text{-}5)$$

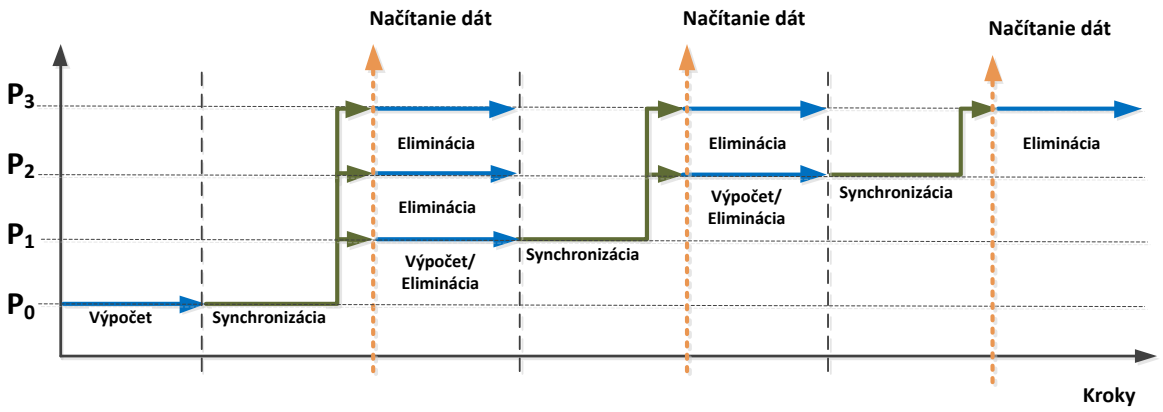
6.2 Paralelné algoritmy GEM

Pri návrhu paralelných algoritmov sa vo väčšine prípadov vychádza so sekvenčného algoritmu. Sekvenčný algoritmus by mal byť čo najefektívnejší a jeho detailná znalosť je veľmi dôležitá pre jeho efektívnu dekompozíciu na paralelný algoritmus.

Úloha riešenia sústavy lineárnych rovníc je rozdelená na dve časti - transformovanie matice a určenie neznámych. My sa budeme zaoberať iba časťou transformácie, lebo tá ma dominantný vplyv na výkonnosť algoritmu.

6.2.1 Paralelizácia nad spoločnou pamäťou

Pri využití spoločnej pamäti pracuje algoritmus nad celou maticou koeficientov n x n. Je vytvorených viacero procesov, z ktorých každý pracuje iba nad svojou častou matice (blok riadokov). Kroky GEM sú znázornené na Obr. 6-2. Prvý proces vykoná výpočet a synchronizuje zvyšné procesy. Pomocou synchronizácie oznamuje nadriadený proces, že vykonal výpočet a uložené dáta sú platne. Procesy po vykonaní synchronizácie načítajú riadok z pamäte nadriadeného procesu a vykonajú elimináciu a výpočet. Ďalší v poradí nadradený proces synchronizuje ostatné procesy a tak sa pokračuje až do úplnej eliminácie matice.



Obr. 6-2 Paralelný GEM algoritmus nad spoločnou pamäťou

6.2.1.1 Celková doba vykonávania

Pri určení celkovej doby vykonávania paralelného algoritmu nad spoločnou pamäťou, sa analýza zložitosti obmedzuje iba na analýzu výpočtovej časti. Komunikačnú časť môžeme zanedbať, vzhľadom na rýchle komunikačné prepojenie pamäte. Paralelnú zložitosť potom odvodíme nasledovne:

Paralelná Gausova eliminácia približne vykonáva $\frac{1}{p}\sum_{j=1}^{n-1}\sum_{k=j}^{n-1} 1 = \frac{-n+n^2}{2p}$ delení a približne $\frac{1}{P}\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=i}^{n} 2 = \frac{n+3n^2+2n^3}{3p}$ odčítaní a násobení. Pre zjednodušenie predpokladáme, že

všetky aritmetické operácie sú vykonávane v jednotkovom čase. Výpočtovú zložitosť môžeme potom zapísať ako

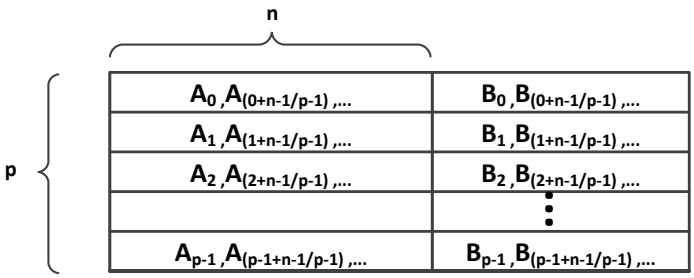
$$T(s,p)_{comp} = t_c\left(\frac{-n+n^2}{2p} + \frac{n+3n^2+2n^3}{3p}\right) \qquad (6\text{-}6)$$

6.2.2 Paralelizácia nad distribuovanou pamäťou

Pri paralelizácii sekvenčného algoritmu je dôležité zvoliť správnu dekompozičnú stratégiu. Ďalej je dôležité zvoliť výhodný model komunikácie. Komunikácia prebieha buď synchrónne alebo asynchrónne.

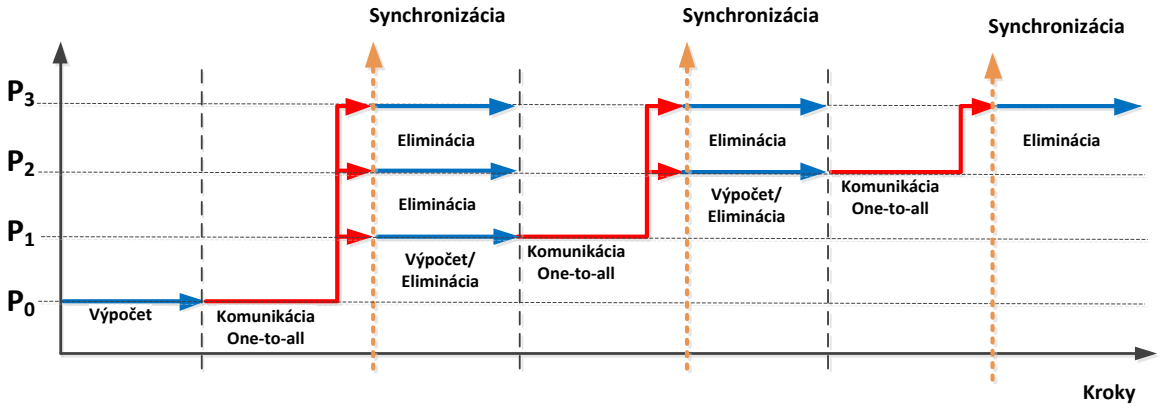
6.2.2.1 Synchrónna implementácia

Pre jednoduchosť je zvolená riadková cyklická dekompozícia. Každému výpočtovému uzlu sú pridelené cyklicky riadky matice koeficientov **A** a prvoky **b** vektora pravých strán. Takto je celá matica **A** o veľkosti *n x n* a vektor **b** o veľkosti *n* rozdelená medzi p procesov, pričom každý proces má pridelených $\frac{n}{p}$ riadkov.



Obr.6-3 Grafické znázornenie cyklickej dekompozície n x n matice na P_{m-1} procesov

Tento paralelný GEM algoritmus vychádza so sekvenčnej verzie, ktorá je doplnená o komunikačnú časť, ktorá má za úlohu odoslať aktuálne spracovávaný riadok zvyšným procesom, ktoré pomocou neho vykonajú elimináciu na svojej časti matice.



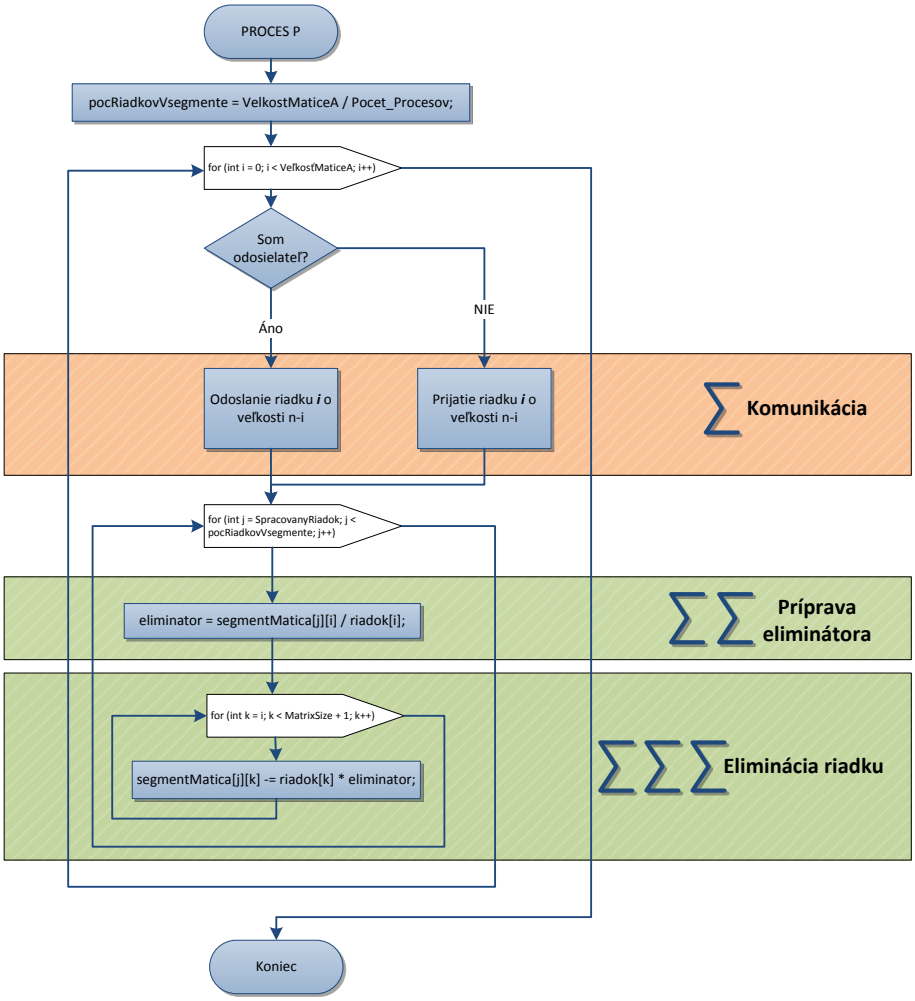
Obr.6-4 Synchrónna paralelná implementácia GEM

6.2.2.1.1 Analýza paralelného algoritmu

6.2.2.1.1.1 Celková doba vykonávania

Celková doba vykonávania paralelného algoritmu je súčet komunikačnej zložitosti a výpočtovej zložitosti

$$T(s,p)_{par}=t_c(\frac{-n+n^2}{2p}+\frac{n+3n^2+2n^3}{3p})+\frac{1}{2}n(2t_s+t_w+nt_w)\log_2 p \qquad (6\text{-}7)$$



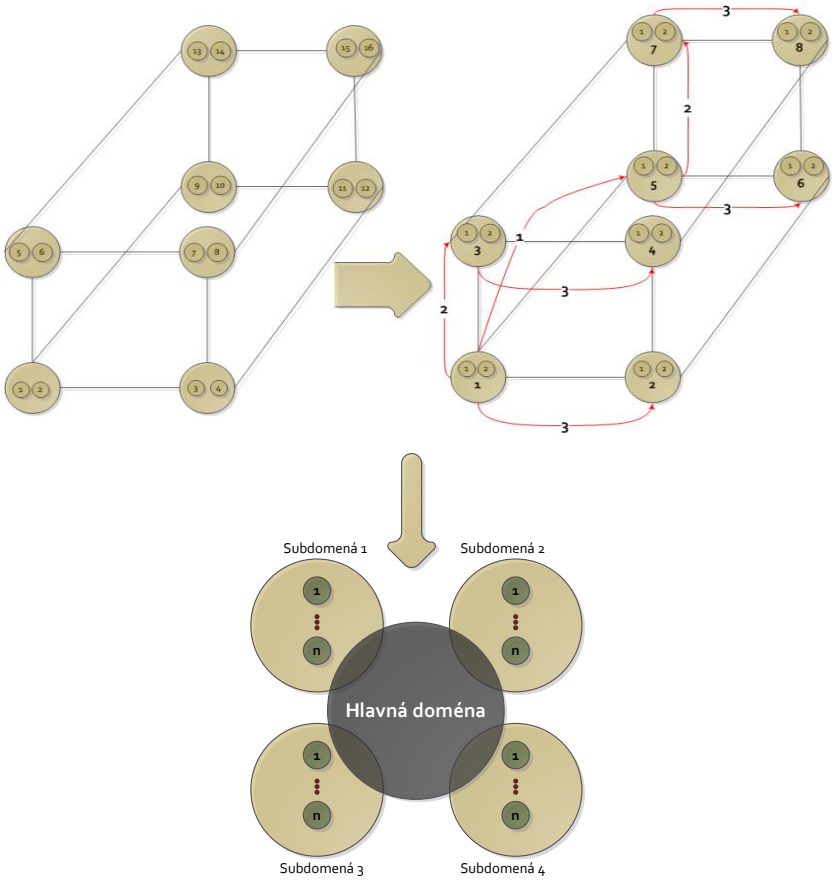
Obr. 6-5 Vývojový diagram paralelného algoritmu GEM s naznačením výpočtovej a komunikačnej zložitosti

6.2.2.1.2 Úprava broadcast komunikačného algoritmu na zlepšenie synchrónnej verzie GEM algoritmu pre vykonávanie vo viacjadrovom prostredí

Novodobý trend SMP počítačov s n-integrovanými výpočtovými jadrami predstavuje budúcnosť paralelných systémov. Dnešné viacjadrové pracovné stanice bežne obsahujú 4, 8, 16 a viac výpočtových jadier. Pre efektívne využitie NOW alebo GRID paralelných počítačov tvorených práve takýmito SMP počítačmi je potrebné prispôsobiť existujúce softvérové prostriedky tak, aby zohľadňovali charakteristiky takéhoto systému.

Z pohľadu optimalizácie GEM synchrónneho algoritmu treba klásť dôraz na zefektívnenie medziprocesnej komunikácie, ktorá je jedným z hlavných limitujúcich faktorov.

Na základe doménového rozdelenia Obr. 7-13 môžeme komunikáciu rozdeliť na vnútrouzlovú (internú), teda komunikácia v rámci jednej subdomény a komunikáciu mimouzlovú (medzipočítačovú) a teda komunikácia v rámci hlavnej domény.



Obr. 6-6 Rozdelenie hlavnej komunikačnej domény na subdomény

Pri časovej analýze daného algoritmu vychádzame z obrázku Obr. 6-7 z neho vyplýva, že odoslanie jednej správy všetkým prijímateľom sa deje v troch krokoch, ktoré sa vykonávajú sekvenčne. Na začiatku chceme odoslať správu o veľkosti *m (p x j)* procesom, kde *p* je počet SMP počítačov s viacerými jadrami (procesormi) a j je počet týchto jadier (procesorov). Pre modelovanie komunikácie bod-bod medzi počítačmi (uzlami) použijeme rovnicu (3-2) a pre komunikáciu medzi jadrami (procesormi) v rámci jedného počítača (uzla) použijeme rovnicu (3-3). Potom pre krok 1 platí rovnica

$$T_1 = (t_{sc} + t_{wc}m)j \qquad (6\text{-}8)$$

, táto rovnica popisuje sekvenčné odoslanie jednej správy všetkým procesom v rámci jedného počítača (uzla).

Pre krok 2 platí rovnica

$$T_2 = (t_s + t_w m)\log_2 p \qquad (6\text{-}9)$$

, táto rovnica popisuje štandardný broadcast ako je popísaný v kapitole 5.3.1. V tomto kroku sa prenesie správa medzi všetkými počítačmi (uzlami) v hlavnej komunikačnej doméne.

Pre krok 3 platí rovnica

$$T_2 = (t_{sc} + t_{wc}m)n \qquad (6\text{-}10)$$

, v tomto kroku sa prenesie správa vo vnútri všetkých zvyšných uzlov jednotlivým jadrám (procesorom). Toto odosielanie sa deje paralelne a preto sa do celkového času započíta iba čas prenesenia správy v rámci jedného počítača. Celkový čas prenesenia jednej správy od jedného procesu všetkým je daný súčtom parciálnych časov a platí

$$\begin{aligned} T = T_1 + T_2 + T_3 T &= (t_{sc} + t_{wc}m) * (n) + (t_s + t_w m)\log_2 p + (t_{sc} + t_{wc}m)(n)T \\ &= 2 * (t_{sc} + t_{wc}m) * (n) + (t_s + t_w m)\log_2 p \end{aligned} \qquad (6\text{-}11)$$

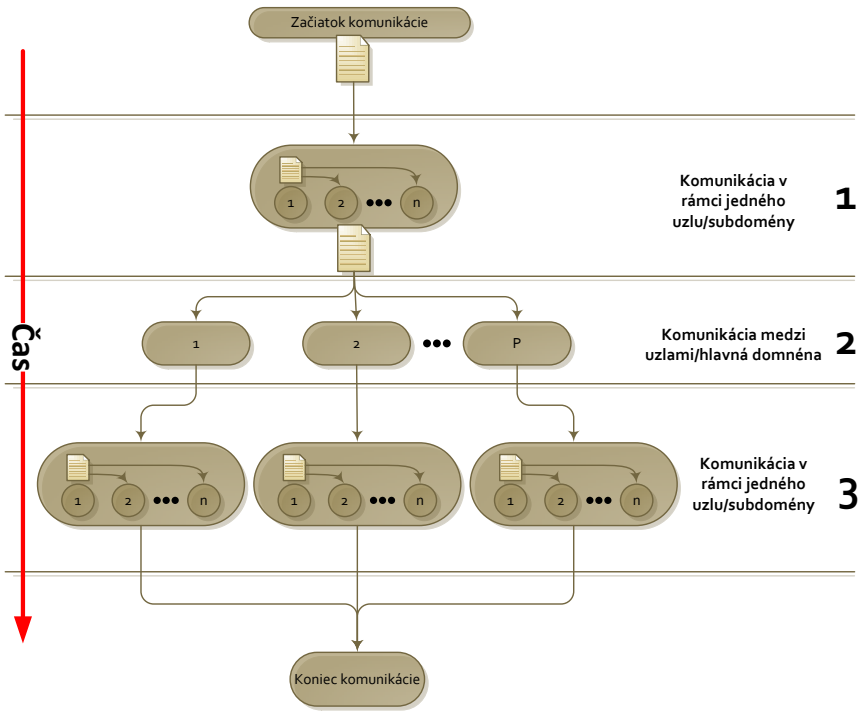
Za predpokladu, že komunikácia vo vnútri počítača (uzla) je rádovo rýchlejšia ako komunikácia medzi počítačmi (uzlami) t.j. $t_{sc} \ll t_s$ a $t_{wc} \ll t_w$, je výsledná rovnica zjednodušená iba na prenos správy medzi počítačmi (uzlami) v hlavnej doméne

$$T = (t_s + t_w m)\log_2 p \qquad (6\text{-}12)$$

, za tohto predpokladu platí pre časovú náročnosť synchrónneho GEM paralelného algoritmu táto rovnica

$$T(s,p)_{par} = t_c\left(\frac{-n + n^2}{2p} + \frac{n + 3n^2 + 2n^3}{3p}\right) + \frac{1}{2}n(2t_s + t_w + nt_w)\log_2 p_{smp} \qquad (6\text{-}13)$$

, kde p je celkový počet procesorov (jadier) paralelného systému a p_{smp} je počet samotných SMP počítačov tvoriacich paralelný počítač. Z daných vzťahov vyplýva, že čím viac budú paralelné počítače tvorené SMP počítačmi s viacerými jadrami tým bude efektivita synchrónneho GEM paralelného algoritmu lepšia.



Obr. 6-7 Kroky navrhnutého subdoménového broadcast komunikačného algoritmu

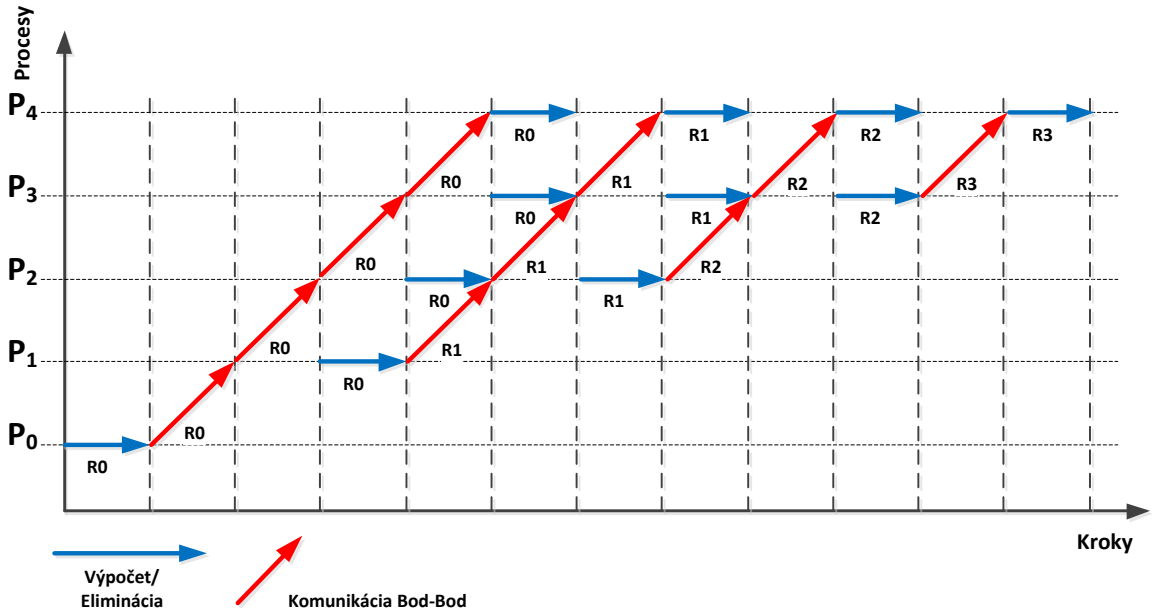
6.2.2.2 Asynchrónna implementácia

Pri synchrónnom algoritme, každý proces v rovnakom časovom okamihu vykonáva rovnaký krok algoritmu (výpočet, komunikácia alebo eliminácia). Do ďalšieho kroku procesy prejdú až keď všetky procesy ukončia predchádzajúci krok. Efektívnejší algoritmus dosiahneme ak procesy budú pracovať asynchrónne (zreťazene).

Pri asynchrónnom GEM každý proces nezávisle vykonáva operácie až kým celý algoritmus nevykoná všetkých n interácií potrebných na transformáciu matice **A** na hornú trojuholníkovú maticu. Procesy vykonávajú nasledovné kroky:

1. Ak má proces nejaké dáta na odoslanie, tak ich odošle konkrétnemu procesu
2. Ak má proces vhodné dáta na výpočet, tak vykoná výpočet
3. V ostatných prípadoch proces čaká na prijatie dát, aby mohol vykonať predchádzajúce dva kroky.

Takto realizovaný algoritmus (viď. Obr.6-8) je optimálny s prihliadnutím na jeho sekvenčný ekvivalent. Pre implementáciu daného algoritmu využívame neblokovacie metódy komunikácie a procesy sú logicky usporiadané do lineárneho zreťazeného zoznamu. Komunikácia prebieha spôsobom podobným one-to-all pre lineárny zreťazený zoznam. Niektoré implementácie MPI ponúkajú neblokovacie kolektívne operácie. V tom prípade sa môže na implementáciu použiť All-to-one (broadcast) komunikačná metóda.



Obr.6-8 Asynchrónna paralelná implementácia GEM

6.2.2.2.1 Úprava broadcast komunikačného algoritmu na transformáciu synchrónnej verzie GEM algoritmu na asynchrónnu

Tento komunikačný algoritmus vychádza z predchádzajúceho algoritmu, ktorý bol popísaný v časti 6.2.2.1.2. Algoritmus rozdeľuje jednu komunikačnú doménu na subdomeny na základe fyzickej štruktúry paralelného počítača, t.j. podľa počtu a štruktúry SMP počítačov.

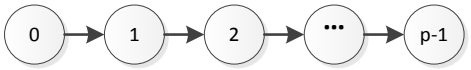
Pri časovej analýze vyplýva, že odoslanie jednej správy všetkým prijímateľom sa deje v troch krokoch, ktoré sa vykonávajú sekvenčne. Na začiatku chceme odoslať správu o veľkosti M $p \times n$ procesom, kde p je počet SMP počítačov s viacerými jadrami (procesormi) a n je počet týchto

jadier (procesorov). Pre modelovanie komunikácie bod-bod medzi počítačmi (uzlami) použijeme rovnicu (3-2) a pre komunikáciu medzi jadrami (procesormi) v rámci jedného počítača (uzla) použijeme rovnicu (3-3). Potom pre krok 1 a krok 2 platí rovnica

$$T_1 = T_2 = (t_{sc} + t_{wc}M)(n) \qquad (6\text{-}14)$$

, táto rovnica popisuje sekvenčné odoslanie jednej správy všetkým procesom v rámci jedného počítača (uzla).

Počítače (uzly) komunikujúce v kroku 2, sú logicky usporiadané do zreťazeného lineárneho stromu ako je znázornené na obrázku Obr. 6-9.



Obr. 6-9 Počítače usporiadané do zreťazeného lineárneho stromu

Správa je rozdelená na menšie správy, ktoré sú postupne odosielané a celým lineárnym stromom prechádzajú zreťazene. Pre časovú analýzu algoritmu ho rozdelíme na dva kroky. V prvom kroku sa odosiela prvá časť správy (paket) lineárnym stromom, až kým nedorazí k poslednému uzlu. Tomuto kroku hovoríme plnenie zreťazeného stromu a je znázornené na obrázku Obr. 6-10, kde *p* je počet komunikujúcich procesov a *m* je veľkosť paketu. Po naplnení lineárneho stromu prechádzajú zvyšné časti správy celým stromom až k poslednému uzlu ako je znázornené na obrázku Obr. 6-11, kde *X* je počet paketov na ktoré bola správa o veľkosti *M* rozdelená a *m* je veľkosť jedného paketu. Na konci tohto procesu majú všetky uzly svoju kópiu odosielanej správy.

Celkový čas prenosu broadcastu pomocou algoritmu lineárneho stromu je súčet času potrebného na prvotné naplnene lineárneho stromu a času odoslanie zvyšných paketov

$$T_2 = (p-1)(t_s + t_w m) + (X-1)(t_w m) \qquad (6\text{-}15)$$

, kde *p* je počet procesov (uzlov), ktoré navzájom komunikujú, *m* je veľkosť jedného paketu a *X* je počet paketov ,na ktoré bola správa o veľkosti *M* rozdelená. $X = \frac{M}{m}$

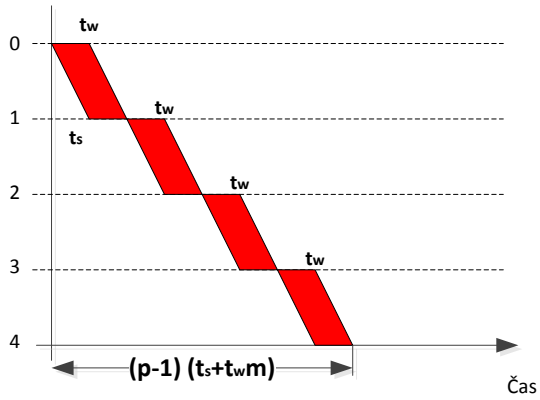
Celkový čas prenesenia jednej správy od jedného procesu všetkým procesom je daný súčtom parciálnych časov a platí

$$T = T_1 + T_2 + T_3$$

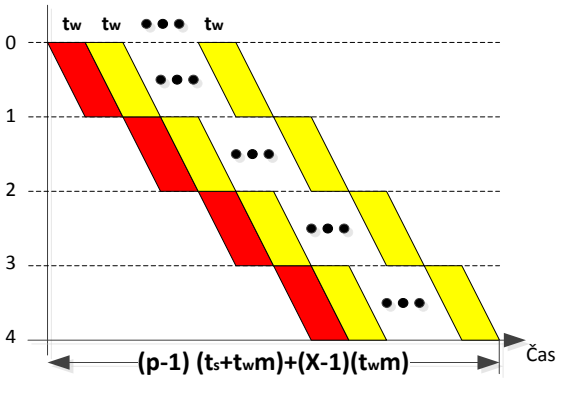
$$\begin{aligned} T &= (t_{sc} + t_{wc}M)(n) + (p-1)(t_s + t_w m) + (X-1)(t_w m) + (t_{sc} + t_{wc}M)(n)T \\ &= 2(t_{sc} + t_{wc}M)(n) + (p-1)(t_s + t_w m) + (X-1)(t_w m) \end{aligned} \qquad (6\text{-}16)$$

Za predpokladu, že komunikácia vo vnútri počítača (uzla) je rádovo rýchlejšia ako komunikácia medzi počítačmi (uzlami) t.j. $t_{sc} \ll t_s \; a \; t_{wc} \ll t_w$, je výsledná rovnica zjednodušená iba na prenos správy medzi počítačmi (uzlami) v hlavnej doméne

$$T = (p-1)(t_s + t_w m) + (X-1)(t_w m) \qquad (6\text{-}17)$$



Obr. 6-10 Naplnenie lineárneho stromu – čas prvého kroku



Obr. 6-11 Odoslanie správy cez zreťazený lineárny strom

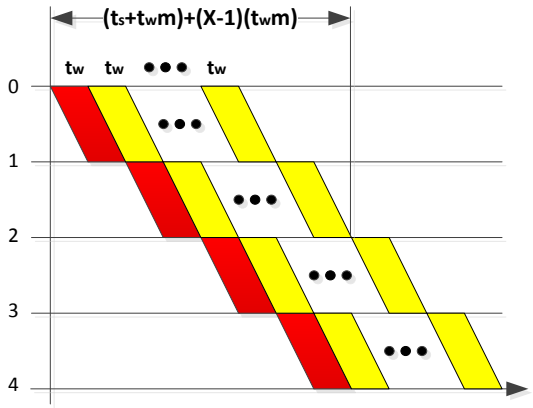
6.2.2.2.2 Analýza GEM asynchrónneho algoritmu

6.2.2.2.2.1 Analýza výpočtovej zložitosti

Paralelná Gausova eliminácia približne vykonáva $\frac{1}{p}\sum_{j=1}^{n-1}\sum_{k=j}^{n-1} 1 = \frac{-n+n^2}{2p}$ delení a približne $\frac{1}{P}\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=i}^{n} 2 = \frac{n+3n^2+2n^3}{3p}$ odčítaní a násobení. Pre zjednodušenie predpokladáme, že všetky aritmetické operácie sú vykonávane v jednotkovom čase. Výpočtovú zložitosť môžeme potom zapísať ako

$$T(s,p)_{comp} = t_c\left(\frac{-n+n^2}{2p} + \frac{n+3n^2+2n^3}{3p}\right) \qquad (6\text{-}18)$$

, kde konštanta t_c predstavuje priemerný čas vykonania jednej operácie.



Obr. 6-12 Odoslanie správy nasledovnému procesu

6.2.2.2.2.2 Analýza komunikačnej zložitosti

Pri asynchrónnom GEM každý proces nezávisle vykonáva operácie až kým celý algoritmus nevykoná všetkých n interácií potrebných na transformáciu matice **A** na hornú trojuholníkovú maticu. To znamená, že po každej eliminácii nastáva komunikácia, pri ktorej sa odošle vypočítaný riadok

nasledujúcemu procesu a ten ho následne spracuje. Procesy sú usporiadané za sebou lineárnym spôsobom. Pri komunikácii s využitím zreťazenej broadcast funkcia je čas potrebný na odoslanie správy nasledujúcemu procesu odvodený z obrázku Obr. 6-12.

Čas prenosu správy je súčet času potrebného na prvotné naplnene lineárneho stromu a času odoslania zvyšných paketov.

$$T = (p-1)(t_s + t_w m) + (X-1)(t_w m) \qquad (6\text{-}19)$$

, kde *p* je počet počítačov (uzlov), ktoré navzájom komunikujú, m je veľkosť jedného paketu a X je počet paketov na ktoré bola správa o veľkosti n rozdelená. $X = \frac{n}{m}$

$$T = (t_s + t_w n) \qquad (6\text{-}20)$$

$$T(s,p)_{comm} = \sum_{i=0}^{n-1} \left(t_s + t_w(n-i)\right) = nt_s + \frac{\mathrm{n}t_w}{2} + \frac{n^2 t_w}{2} \qquad (6\text{-}21)$$

,kde t_s je čas inicializácie prenosu a t_w je doba prenosu jedného slova.

6.2.2.2.2.3 Celková doba vykonávania

Celková doba vykonávania paralelného algoritmu je súčet komunikačnej zložitosti a výpočtovej zložitosti

$$T(s,p)_{par}=t_c\left(\frac{-n+n^2}{2p} + \frac{n+3n^2+2n^3}{3p}\right) + nt_s + \frac{\mathrm{n}t_w}{2} + \frac{n^2 t_w}{2} \qquad (6\text{-}22)$$

6.2.2.2.3 Analýza algoritmu pre vykonávanie vo viacjadrovom (viacprocesorovom) prostredí

Pri analyzovaní algoritmu predpokladáme, že paralelný počítač je zostavený z rovnakých SMP počítačov, ktoré majú rovnaký počet jadier (procesorov) a ich počet je označený m_{core}. Procesy sú prideľované blokovo jednotlivým procesorom a jadrám **Chyba! Nenalezen zdroj odkazů.**. V takomto prípade môžeme rozdeliť komunikáciu, ktorá sa vykonáva medzi jadrami (procesormi) počítača a medzi počítačmi navzájom. Komunikáciu asynchrónneho GEM algoritmu, tak ako je popísaná rovnicou (6-21), môžeme tiež rozdeliť na vnútornú a vonkajšiu komunikáciu. Pri predpoklade, že procesy sú rozdelené blokovo-lineárnym spôsobom, celkový čas komunikácie algoritmu môžeme rozdeliť nasledovne

$$\begin{aligned} T_{comm} &= \sum_{i=1}^{n} (\mathrm{t}_{s1} + \mathrm{t}_{w1}\, i) - \sum_{i=1}^{\frac{n}{m_{core}}} (\mathrm{t}_{s1} + \mathrm{t}_{w1}\, i * m_{core}) + \sum_{i=1}^{\frac{n}{m_{core}}} (\mathrm{t}_{s2} + \mathrm{t}_{w2}\, i * m_{core}) \\ &= n\mathrm{t}_{s1} - \frac{n\mathrm{t}_{s1}}{m_{core}} + \frac{n\ \mathrm{t}_{s2}}{m_{core}} + \frac{n^2 \mathrm{t}_{w1}}{2} - \frac{n^2 \mathrm{t}_{w1}}{2m_{core}} + \frac{\mathrm{n}t_{w2}}{2} + \frac{n^2 \mathrm{t}_{w2}}{2m_{core}} \end{aligned} \qquad (6\text{-}23)$$

, kde *n* je veľkosť matice, t_{s1}, t_{s2} je čas inicializácie prenosu a t_{w1}, t_{w2} je doba prenosu jedného slova, pričom t_{s1}, t_{w1} sú konštanty určujúce prenos v rámci počítača a t_{s2}, t_{w2} sú konštanty určujúce prenos medzi počítačmi. m_{core} je počet jadier (procesorov) v jednom počítači. Pri predpoklade, že komunikácia v rámci počítača je rádovo rýchlejšia ako komunikácia medzi počítačmi

t.j. $t_{s1}, t_{w1} \ll t_{s2}, t_{w2}$ sa môže komunikácia v rámci počítača zanedbať a celkový čas komunikácie bude určený komunikáciou medzi počítačmi

$$T(s,p)_{comm} = \frac{n\,\mathrm{t}_{s2}}{m_{core}} + \frac{\mathrm{nt}_{w2}}{2} + \frac{n^2\mathrm{t}_{w2}}{2m_{core}} \qquad (6\text{-}24)$$

6.2.2.3 Hybridný algoritmus

Tento algoritmus pracuje nad distribuovanou a spoločnou pamäťou. Takáto kombinácia vzniká využitím technológii pre komunikáciu s využitím posielania správ MPI a komunikáciu cez spoločnú pamäť openMP. Pri návrhu daného algoritmu vychádzame z asynchrónnej verzie GEM algoritmu nad distribuovanou pamäťou. Úprava pôvodného algoritmu spočíva v rozdelení cyklov pre elimináciu riadkov matice na vlákna, ktoré budú vykonávané na jadrách SMP počítača.

6.2.2.3.1 Celková doba vykonávania algoritmu

Celková doba vykonávania paralelného algoritmu je súčet komunikačnej zložitosti a výpočtovej zložitosti a času inicializácie

$$T(s,p)_{par} = \frac{t_c}{p_{core}}\left(\frac{-n+n^2}{2p} + \frac{n+3n^2+2n^3}{3p}\right) + \frac{n\,\mathrm{t}_s}{p} + \frac{\mathrm{nt}_w}{2} + \frac{n^2\mathrm{t}_{w2}}{2p} + \frac{1}{2}n(n-1)t_{init}\,p_{core} \qquad (6\text{-}25)$$

7 Metodika merania

Podľa spôsobu merania môžeme metodiky rozdeliť na priame a nepriame. Pre empirické meranie výkonnosti algoritmov sú vhodné priame metódy pre ich jednoduchú implementáciu.

7.1 Metodika merania sekvenčného algoritmu

Empirickými testami zisťujeme výkonnostné metriky sekvenčného programu. Testy skúmajú dobu vykonávania, ktorá je nutná na spracovanie sekvenčného programu. Každý test sa skladá z piatich relácií, v rámci ktorých zaznamenáme dobu vykonávania sekvenčného programu. Potom vypočítame priemernú dobu vykonania sekvenčného programu.

7.2 Metodika merania paralelného algoritmu

Základným predpokladom pre meranie komplexných metrík výkonnosti paralelného systému je vývoj konkrétneho paralelného programu pre danú architektúru paralelného počítača s dostupným programovým vybavením. GEM paralelné algoritmy boli implementované pre architektúru s distribuovanou pamäťou za pomoci technológie posielania správ (MPI) a pre architektúru so spoločnou pamäťou za pomoci štandardu openMP.

7.3 Merania na architektúre so spoločnou pamäťou

Analogicky ako pre meranie doby vykonania sekvenčných algoritmov, bola pre merania na architektúrach so spoločnou pamäťou využitá analogická metodika. Dobu vykonania algoritmu sme merali pomocou vstavanej funkcie omp_get_wtime() z knižnice OpenMP, ktorá vráti počet milisekúnd od určeného bodu. Vývojový diagram zobrazuje metodiku priameho merania.

7.4 Merania na architektúre s distribuovanou pamäťou

Určenie metrík paralelných algoritmov na architektúrach s distribuovanou pamäťou je založené na meraní celkovej exekučnej doby algoritmu. Priame merania na architektúrach s distribuovanou pamäťou sú založené na meraní celkovej doby vykonania paralelného algorit-

mu. Pri analýze paralelného algoritmu je potrebné brať do úvahy režijne oneskorenia (náklady), ktoré sú potrebné na zabezpečenie komunikácie medzi procesmi. Jednotlivé zložky je potrebné merať oddelene

$$T(s,p)_{total} = T(s,p)_{comm} + T(s,p)_{comp}$$

Režijné oneskorenia (náklady) tvorí zložka doby potrebnej na komunikáciu medzi procesmi *T(s,p)comm a* výpočtové náklady tvorí zložka výpočtu *T(s,p)comp*. Algoritmus sa potom delí na časti realizujúce komunikáciu a časti realizujúce samotný výpočet.

Meranie sa vykonáva pre rôznu veľkosť problému a rôzny počet procesov. Exekučný čas sa meral za pomoci vstavanej funkcie MPI_Wtime(), ktorá je obsiahnutá v knižnici MPI.

Meranie sa opakuje pre rôzne počty procesov ako aj pre rôznu veľkosť problému. Dobu trvania algoritmu sme merali pomocou funkcie MPI_Wtime(), ktorá je štandardnou funkciou knižnice MPI. Táto funkcia vracia počet milisekúnd od určeného bodu.

7.5 Použitá architektúra

Pre architektúru so spoločnou pamäťou bol použitý štandard openMP. Všetky algoritmy boli implementované v jazyku C++. Ako kompilátor bol použitý GNU GCC vo verzii 4.6. Meranie algoritmu využívajúceho prostredie so spoločnou pamäťou prebehlo na viacjadrovom serveri s dvoma procesormi Intel Xeon.

Merania algoritmov pre prostredie s distribuovanou pamäťou boli vykonané na architektúre typu NOW, ktorá pozostávala zo ôsmych SMP počítačov, ktoré sú prepojené navzájom cez eternetový gigabitový prepínač (switch) fy 3Com. Parametre počítačov sú uvedené v tabuľke Tab. 7-1. Na pracovných staniciach bol nainštalovaný 64bitový Ubuntu Linux 12.10. Ako MPI (Message passing interface) implementácia bola použitá MPICH2 vo verzii 1.4.0.

Počet uzlov	Procesor	Počet jadier	Pamäť RAM	Sieťová karta
8	quad core q6600 2.4Ghz	4	4GB	1Gb/s

Tab. 7-1 Parametre SMP počítačov

8 Experimentálne výsledky

Uvedenú metodiku merania metrík paralelných algoritmov sme použili na experimentálne získanie výsledkov implementovaných paralelných algoritmov synchrónnej a asynchrónnej GEM verzie nad distribuovanou pamäťou, GEM algoritmu nad spoločnou pamäťou. Experimentálne výsledky sú vyhodnotené z pohľadu výkonnostných kritérií paralelných algoritmov tak ako sú popísané v kapitole 4.

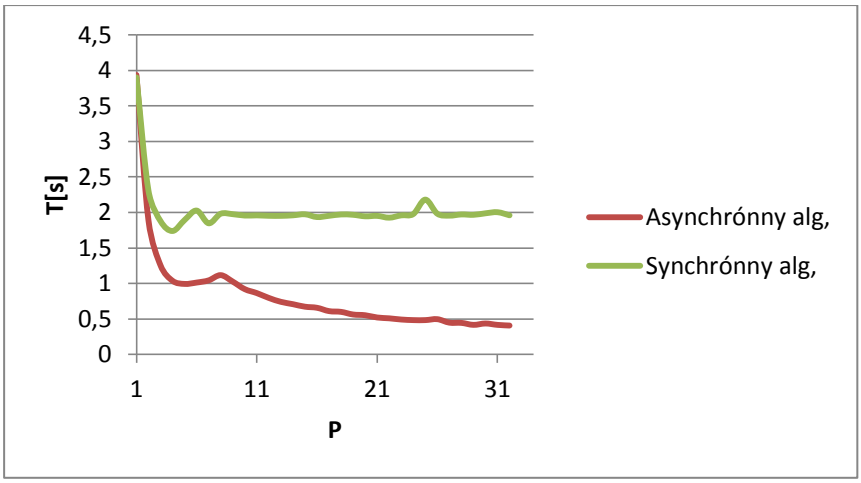
8.1 Merania na architektúrach s distribuovanou pamäťou

8.1.1 Výsledná doba vykonávania

Výsledný exekučný čas paralelných algoritmov sme merali pri konštantnej vstupnej záťaži s premenlivým počtom procesov ako aj pri premenlivej vstupnej záťaži s konštantným počtom procesov. Nasledovné merania sú počítané pre vstupnú záťaž *n=1500,* t.j. vstupná matica je veľkosti 1500x1500. Výpočet sa vykonáva v dvojitej presnosti desatinného čísla (DOUBLE).

8.1.2 Vplyv rastu počtu procesov

Graf Obr. 8-1 ilustruje závislosť exekučného času vykonávania synchrónnej a asynchrónnej verzie GEM algoritmu od počtu procesov, pri konštantnej záťaži.

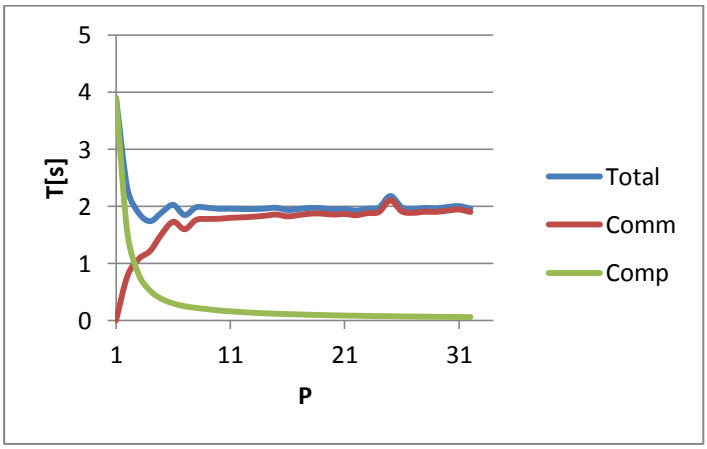


Obr. 8-1 Vplyv zmeny počtu procesov na exekučný čas, n=1500

Z grafu je zrejmé, že doba vykonania s rastúcim počtom procesov klesá. U asynchrónneho algoritmu podľa analýz vykonaných v časti 6.2.2.2.2.2 by mala byť spodná hranica času ohraničená časom prenosu riadku susednému procesu. Naopak synchrónny algoritmus má určitý bod zlomu, keď čas komunikácie presiahne čas výpočtu a ten začne ovplyvňovať celkovú dobu vykonania. Tento bod zlomu môžeme pozorovať aj na zobrazenom grafe, kedy exekučný čas s rastúcim počtom procesorov začína stúpať.

Pre podrobnú analýzu tohto javu je potrebné rozdeliť celkovú dobu vykonávania na parciálne zložky a to čas komunikácie medzi procesmi (t_{comm}) a čas výpočtu (t_{comp}).

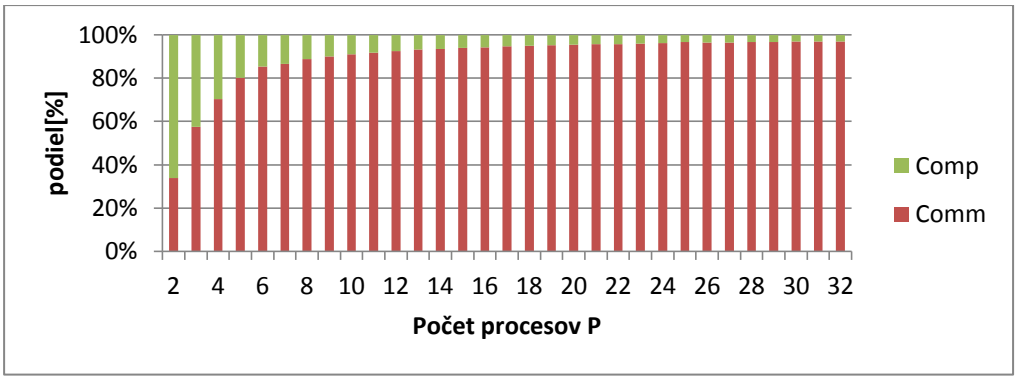
Graf Obr. 8-2 zobrazuje celkovú dobu vykonávania synchrónneho GEM algoritmu rozdelenú na jednotlivé zložky.



Obr. 8-2 Analýza vplyvu výpočtových a komunikačných oneskorení, pre synchrónny GEM alg. pri vstupnej veľkosti problému n = 1500

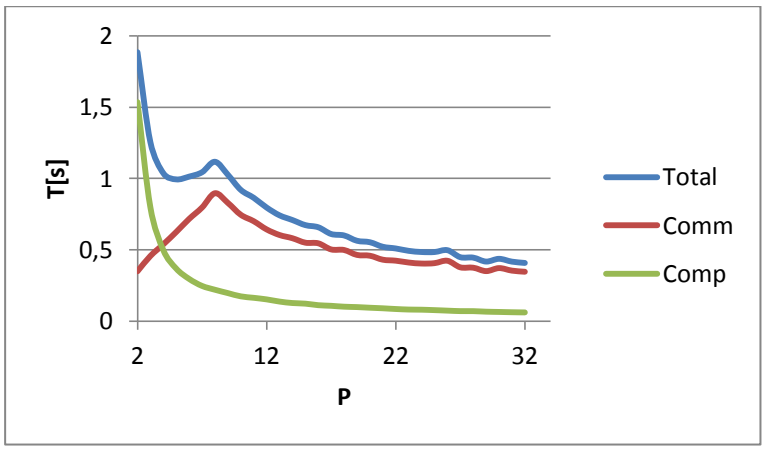
Postupným zväčšovaním počtu procesov zapojených do výpočtu sa čas výpočtu znižuje, ale od určitého bodu zlomu, keď komunikácia dosiahne rýchlejší rast ako je pokles doby výpočtu, doba vykonávania začína na byť ohraničená časom komunikácie. Od tohto bodu začína celková doba

vykonávania narastať. Graf Obr. 8-3 ilustruje percentuálny podiel jednotlivých zložiek exekučného času. Pri porovnaní zložiek vidieť, že komunikácia sa stáva rýchlo dominantnou zložkou.

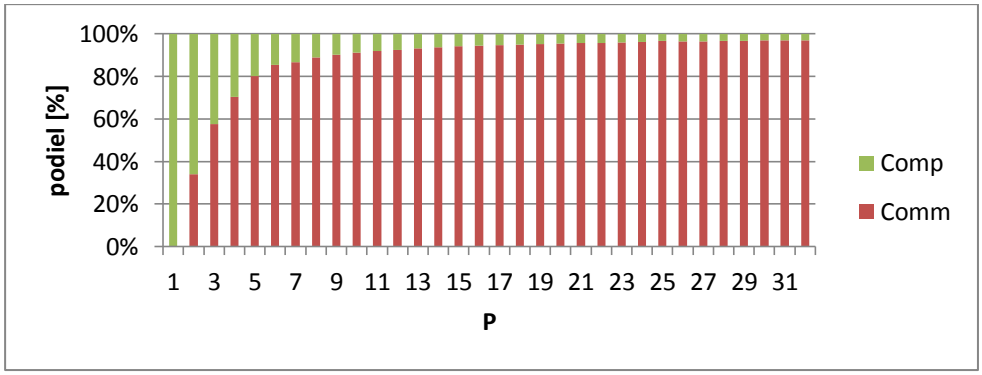


Obr. 8-3 Percentuálny podiel zložiek výpočtu a komunikácie

Graf Obr. 8-4 zobrazuje celkový exekučný čas asynchrónneho GEM algoritmu rozdelený na jednotlivé zložky. Z grafu vidno, že komunikácia pre malý počet procesov stúpa, je to dôsledok nežiadúcej synchronizácie procesov, ktorá je spôsobená tým, že výpočet je natoľko náročný, že nedochádza k reťazeniu výpočtu a procesy sú nútené čakať na platné dáta. Pri náraste počtu procesov sa začína napĺňať reťaz výpočtov a komunikácia sa začína prekrývať s výpočtami. Pri dostatočne veľkom počte procesov sa komunikácia ustáli na konštantnej hodnote.



Obr. 8-4 Analýza vplyvu výpočtových a komunikačných oneskorení na celkový exekučný čas, pre asynchrónny GEM alg. pri vstupnej veľkosti problému n = 1500

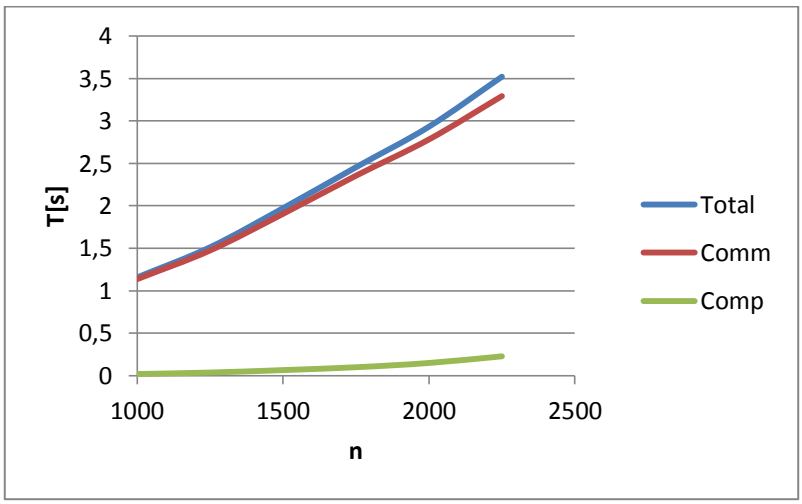


Obr. 8-5 Percentuálny podiel zložiek výpočtu a komunikácie

Graf Obr. 8-5 ilustruje percentuálny podiel jednotlivých zložiek exekučného času. Pri porovnaní zložiek je zrejmé, že komunikácia taktiež dominuje nad výpočtom, ale pomer je priaznivejší ako pri synchrónnom GEM algoritme.

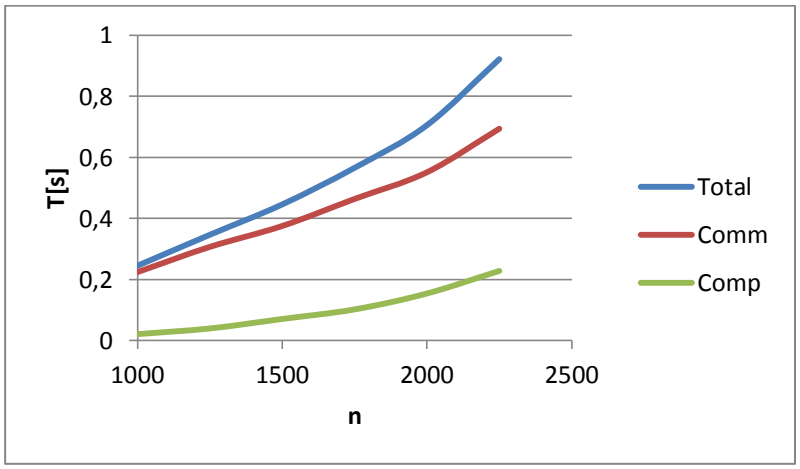
8.1.3 Vplyv rastu záťaže

Graf Obr. 8-6 ilustruje vplyv rastúcej vstupnej záťaže na celkovú dobu vykonávania GEM synchrónneho algoritmu, ako aj na jeho jednotlivé zložky, pri konštantnom počte procesov p=28. Z rastúcim vstupným zaťažením sa zvyšuje čas výpočtu a aj množstvo komunikácie medzi procesmi. Následkom čoho sa zvyšuje celková doba výpočtu. Rast komunikačnej zložky je oveľa väčší ako rast výpočtovej zložky a má hlavný podiel na celkovom raste doby vykonávania algoritmu.



Obr. 8-6 Vplyv výpočtových a komunikačných oneskorení na celkovú dobu vykonania GEM synchrónneho algoritmu

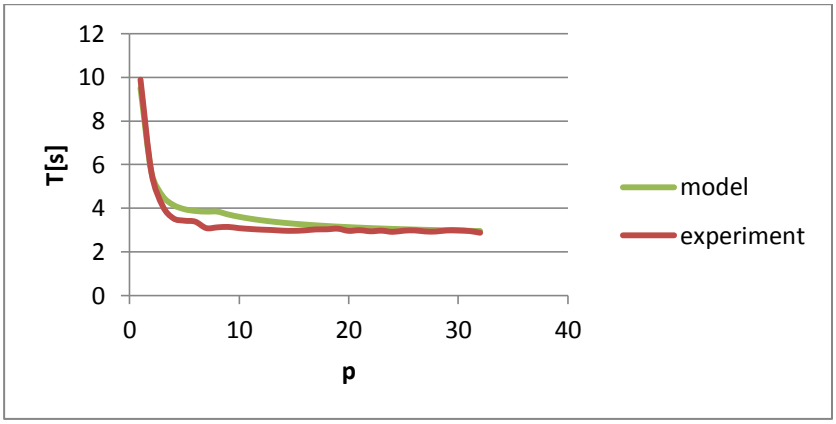
Vplyv rastúcej vstupnej záťaže na dobu vykonávania asynchrónneho GEM paralelného algoritmu je ilustrovaný grafom Obr. 8-7. Z narastajúcim vstupným zaťažením sa zvyšuje čas výpočtu a aj množstvo komunikácie medzi procesmi. Rast komunikačnej zložky je väčší ako rast výpočtovej zložky a má hlavný podiel na celkovom raste doby vykonávania algoritmu. V porovnaní so synchrónnym algoritmom, je pomer medzi komunikačnou zložku a výpočtovou zložkou priaznivejší.



Obr. 8-7 Vplyv výpočtových a komunikačných oneskorení na celkovú dobu vykonania GEM asynchrónneho algoritmu

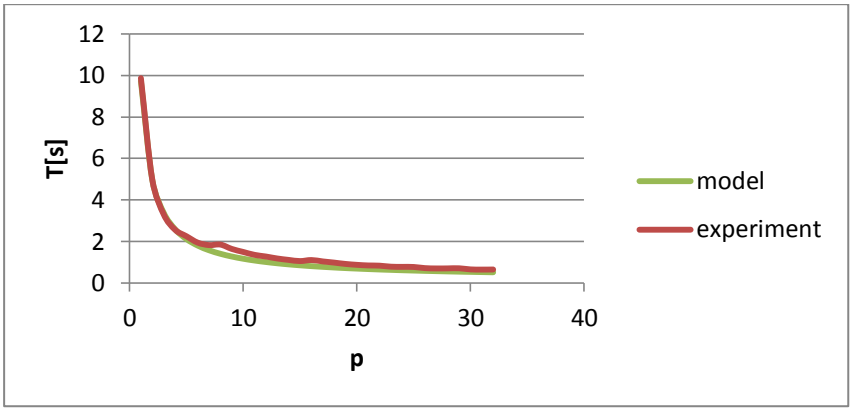
8.1.4 Porovnanie prediktívneho modelu s nameranými hodnotami

V kapitolách 6.2.2.1.1 a 6.2.2.2.2 boli odvodené modely zložitosti pre synchrónny a asynchrónny algoritmus GEM. Na grafe Obr. 8-8 je znázornené porovnanie nameraných výsledkov synchrónneho algoritmu s hodnotami určenými predikčným modelom(6-13), pre vstupnú záťaž *n*=2000 a pri použití konštánt $t_c = 3.56ns, t_s = 84\mu s, t_w = 127ns$.



Obr. 8-8 Synchrónny GEM alg. n=2000

Na grafe je znázornené porovnanie nameraných výsledkov asynchrónneho algoritmu s hodnotami určenými modelom(6-13),(6-13) pre vstupnú záťaž *n*=2000 a pri použití konštánt $t_c = 3.56ns, t_s = 84\mu s, t_w = 127ns$.

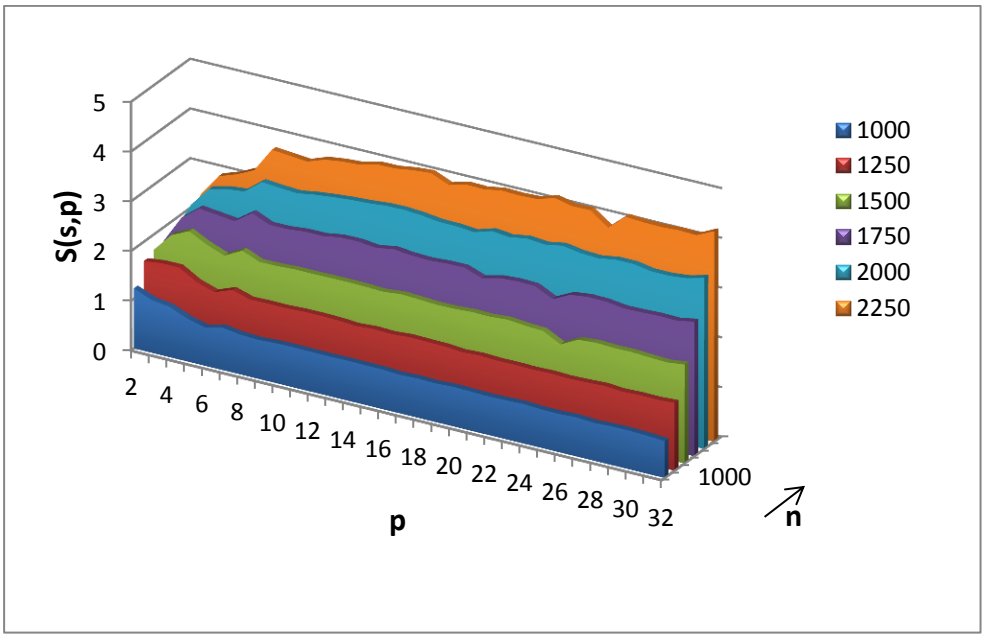


Obr. 8-9 Asynchrónny GEM alg. n=2000

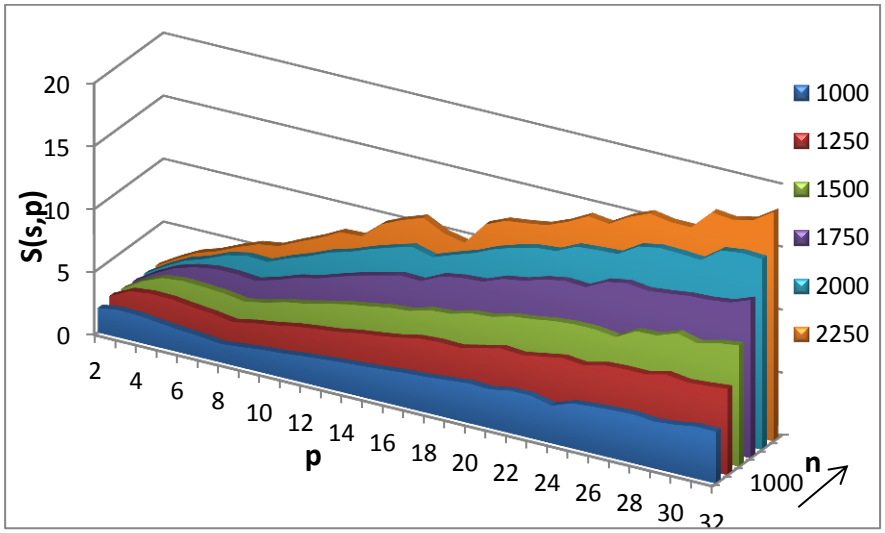
8.1.5 Paralelné zrýchlenie

Na základe predchádzajúcich meraní doby vykonávania algoritmu môžeme odvodiť ich hodnoty paralelného zrýchlenia, ako je popísane v časti 4.1.2.1. Výpočet paralelného zrýchlenia bol vykonaný pomocou rovnice(4-2). V grafe Obr. 8-10 je zobrazená závislosť paralelného zrýchlenia synchrónneho GEM algoritmu od počtu procesov a veľkosti vstupného problému. Z grafu vidno, že rast paralelného zrýchlenia sa už po malom zvýšení počtu procesov zastaví a začína stagnovať.

Na grafe Obr. 8-11 je zobrazená závislosť paralelného zrýchlenia asynchrónneho GEM algoritmu od počtu procesov a veľkosti vstupného problému. Z grafu výplýva, že paralelné zrýchlenie rastie s počtom procesov. Tento rast je sublineárny. Asynchrónny algoritmus je dobre škálovateľný t.j. pri zväčšení vstupnej záťaže sa rast paralelného zrýchlenia zväčší, z čoho vyplýva že daný algoritmus je vhodný aj pre veľké vstupné problémy (záťaže).



Obr. 8-10 Paralelné zrýchlenie synchrónneho algoritmu

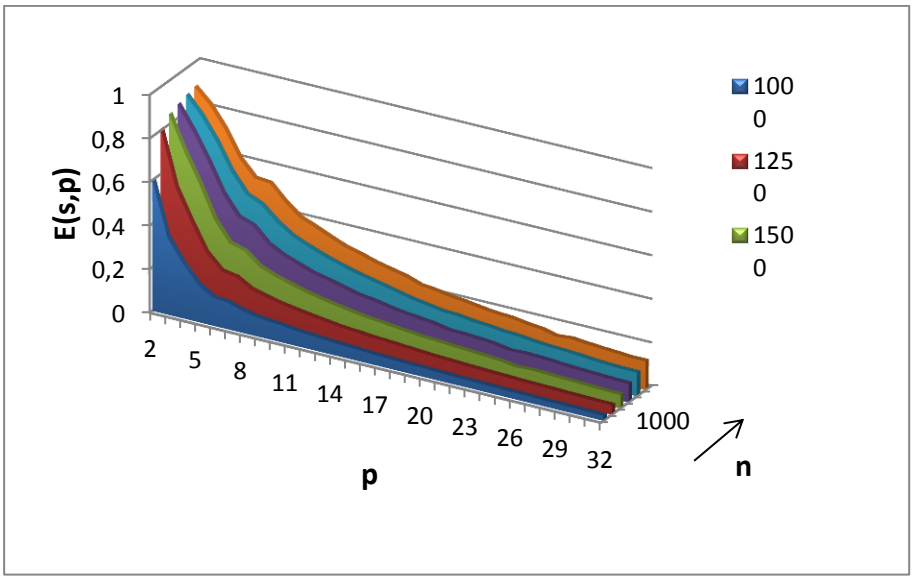


Obr. 8-11 Paralelné zrýchlenie asynchrónneho algoritmu

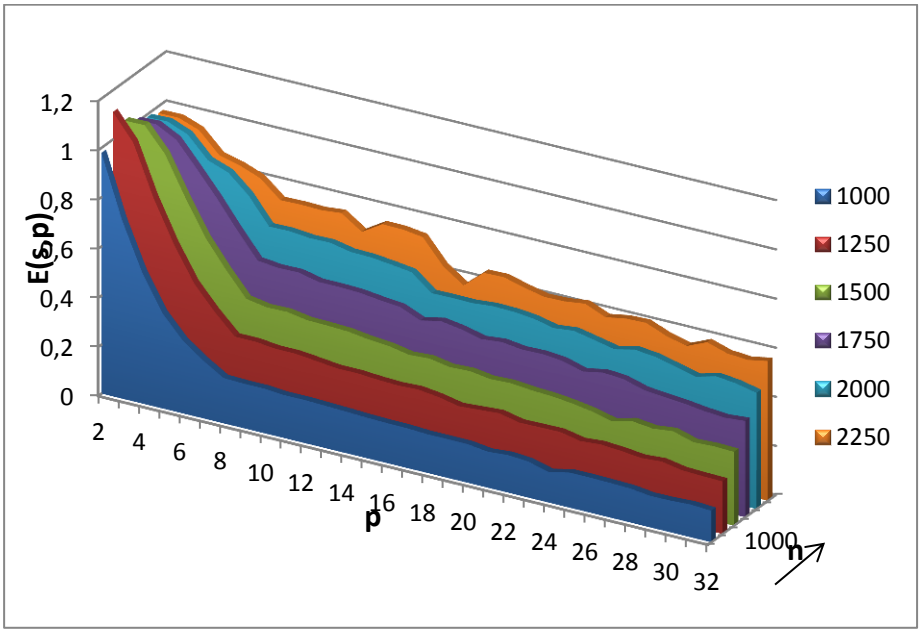
8.1.6 Efektivita využitia výpočtových zdrojov

Efektivita využitia výpočtových zdrojov je dôležitý ukazovateľ pre posúdenie vhodnosti algoritmu pre konkrétnu architektúru paralelného počítača. Efektivita vyjadruje mieru využitia výpočtových kapacít počas výpočtu paralelného programu. Efektivita paralelného programu je odvodená v časti 4.1.2.2. Výpočet efektivity bol vykonaný pomocou rovnice (4-3).

Na grafe Obr. 8-12 je zobrazená závislosť efektivity synchrónneho GEM algoritmu od počtu procesov a veľkosti vstupného problému. Z grafu vyplýva, že efektivita využitia výpočtových zdrojov s rastúcim počtom procesov rýchlo klesá pre celú škálu vstupnej záťaže. Synchrónny algoritmus neefektívne využíva pridelené výpočtové zdroje. Pre lepšiu efektívnosť by bolo potrebné zväčšovať veľkosť vstupnej záťaže a to podľa funkcie izoefektivity.



Obr. 8-12 Efektivita synchrónneho algoritmu



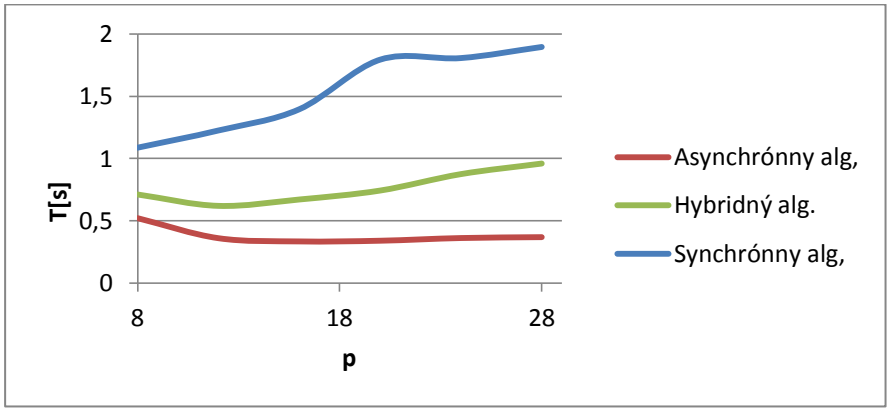
Obr. 8-13 Efektivita asynchrónneho algoritmu

Na grafe Obr. 8-13 je zobrazená závislosť efektivity asynchrónneho GEM algoritmu od počtu procesov a veľkosti vstupného problému. Z grafu vyplýva, že efektivita využitia výpočtových zdrojov s rastúcim počtom procesov klesá, ale už pri malom zväčšení vstupnej záťaže asynchrónny algoritmus dosahuje lepšie hodnoty efektivity ako synchrónna verzia. Pre dosiahnutie požadovanej efektivity pre konkrétnu architektúru paralelného počítača, musíme prispôsobovať veľkosť vstupnej záťaže počtu procesov (procesorov) a to podľa odvodenej funkcie izoefektivity.

8.2 Meranie hybridného algoritmu

Hybridný algoritmus využíva kombináciu prostredia s distribuovanou pamäťou spolu s prostredím so spoločnou pamäťou. Merania sa sústreďujú na porovnanie hybridného algoritmu s asynchrónnym GEM algoritmom vzhľadom na to, že hybridný algoritmus vychádza práve z tohto algoritmu. Merania boli vykonané na sieti NOW tvorenej siedmymi SMP počítačmi, pričom každý obsahoval štvorjadrový procesor Intel Q6600. Každý proces pri hybridnom algoritme bol rozdelený na štyri vlákna. V grafoch je znázornený počet procesov vynásobený počtom jadier

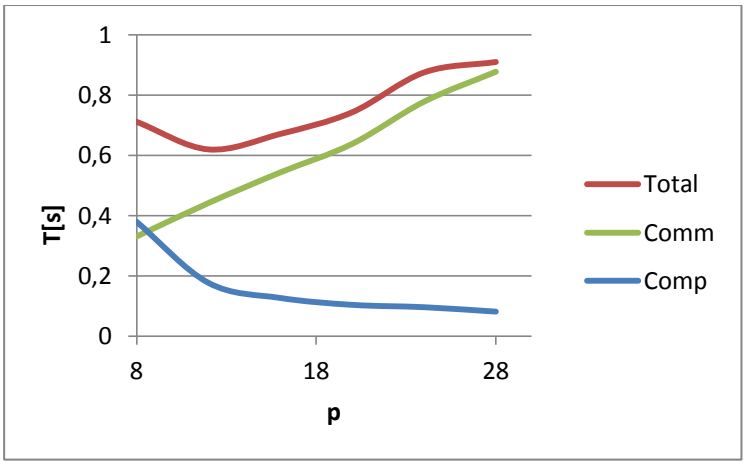
v počítači, aby sme mohli porovnať jednotlivé algoritmy navzájom. Na grafe Obr. 8-14 je znázornený vplyv rastu počtu procesov na celkovú dobu vykonávania algoritmu pri konštantnej vstupnej záťaži n=1500. Hybridný algoritmus vykazuje horšie výsledky ako asynchrónny algoritmus. Spôsobuje to nutnosť inicializácie vlákien pri každom prechode maticou a tiež ich implicitná synchronizácia na konci každého cyklu výpočtu. Z grafu vyplýva, že hybridný algoritmus je efektívnejší ako synchrónny algoritmus. Vzhľadom k tomu, že hybridný algoritmus vychádza z asynchrónneho algoritmu a ten je efektívnejší ako synchrónny, tak je aj hybridný algoritmus efektívnejší ako synchrónny algoritmus.



Obr. 8-14 Vplyv zmeny počtu procesov na dobu výpočtu,n=1500

8.2.1 Vplyv rastu počtu procesov

Pre podrobnú analýzu hybridného algoritmu je potrebné rozdeliť celkovú dobu vykonávania algoritmu na parciálne zložky - čas komunikácie medzi procesmi (t_{comm}), čas výpočtu(t_{comp}) a čas inicializácie vláken. Meranie inicializačnej zložky doby vykonávania hybridného algoritmu nebolo možné vykonať vzhľadom k tomu, že táto zložka je veľmi závislá od operačného systému, ktorý zabezpečuje inicializáciu vláken a ich spustenie na procesore. Jej hodnoty vykazujú veľkú variabilitu a v porovnaní s komunikačnou zložkou sú zanedbateľné. Graf Obr. 8-2 zobrazuje celkovú dobu vykonávania hybridného algoritmu rozdelenú na jednotlivé zložky.

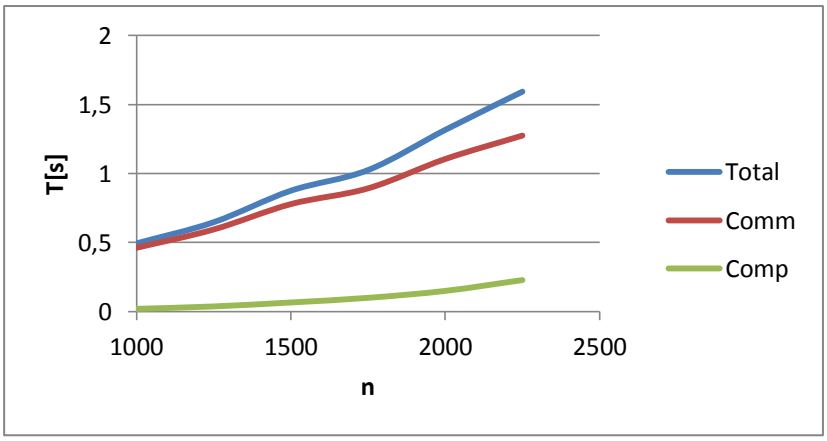


Obr. 8-15 Vplyv výpočtových a komunikačných oneskorení na celkovú dobu vykonania hybridného algoritmu, n = 1500

8.2.2 Vplyv rastu záťaže

Graf Obr. 8-16 ilustruje vplyv rastúcej vstupnej záťaže na celkovú dobu vykonávania hybridného algoritmu, ako aj na jeho jednotlivé zložky pri konštantnom počte procesov p=28. Z rastúcim

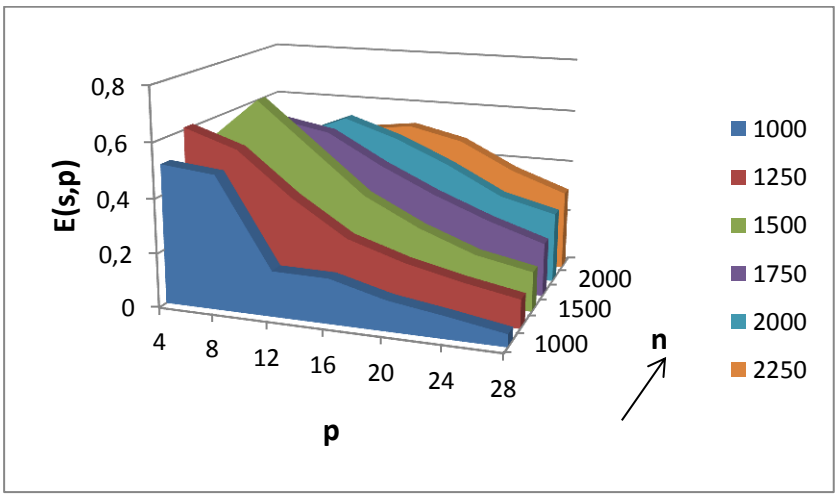
vstupným zaťažením sa zvyšuje čas výpočtu a aj množstvo komunikácie medzi procesmi. V dôsledku čoho sa zvyšuje celková doba výpočtu.



Obr. 8-16 Vplyv výpočtových a komunikačných oneskorení na celkovú dobu vykonania hybridného algoritmu pri rastúcej vstupnej záťaži, p=24

8.2.3 Efektivita využitia výpočtových zdrojov

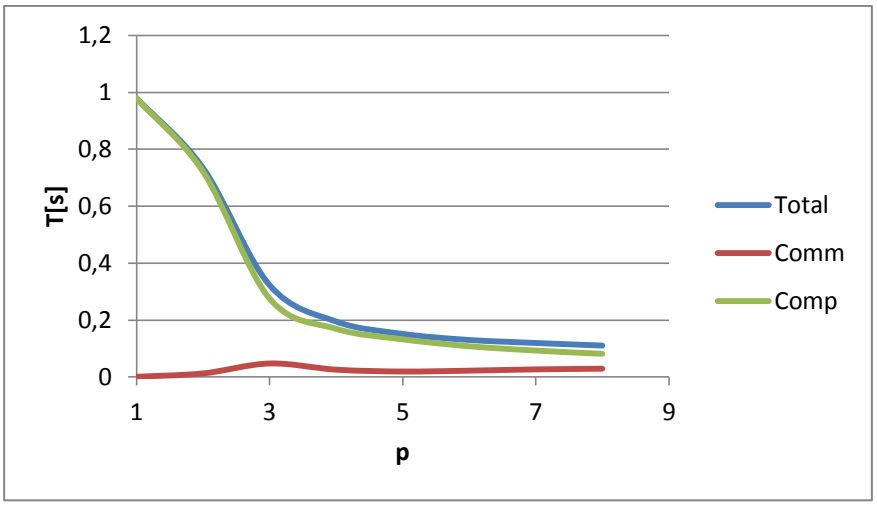
Na grafe Obr. 8-17 je zobrazená závislosť efektivity hybridného al goritmu od počtu procesov a veľkosti vstupného problému. Z grafu vidno, že efektivita využitia výpočtových zdrojov s rastúcim počtom procesov klesá, ale pri zväčšení vstupnej záťaže tento pokles efektivity je menší. V porovnaní s asynchrónnym algoritmom je hybridný algoritmus menej škálovateľný. Je to zapríčinené tým, že pri zlučovaní výpočtových vláken v rámci jedného procesu dochádza k ich synchronizácii, čo má za následok menšie zreťazenie výpočtov.



Obr. 8-17 Efektivita

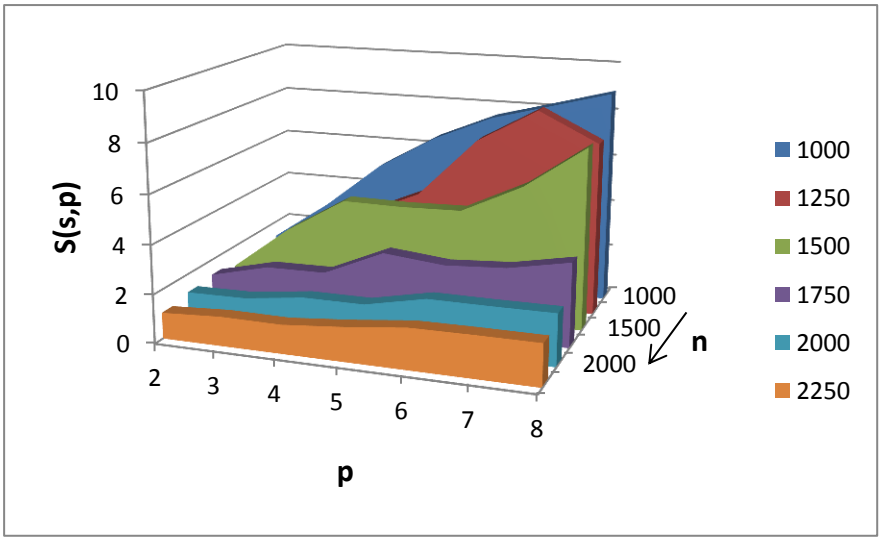
8.3 Merania na spoločnej pamäti

Meranie algoritmu pracujúceho v prostredí so spoločnou pamäťou bolo vykonané na SMP pracovnej stanici fy. HP pozostávajúcej z dvoch procesorov Intel Xeon (každý obsahuje štyri jadrá). Systém disponoval 8 GB operačnej pamäte a bol použitý 64bitový operačný systém Ubuntu Linux vo verzii 12.10. Algoritmus bol paralelizovaný prostredníctvom technológie openMP.



Obr. 8-18 Vplyv výpočtových a komunikačných oneskorení na celkovú dobu vykonania algoritmu, n = 1000

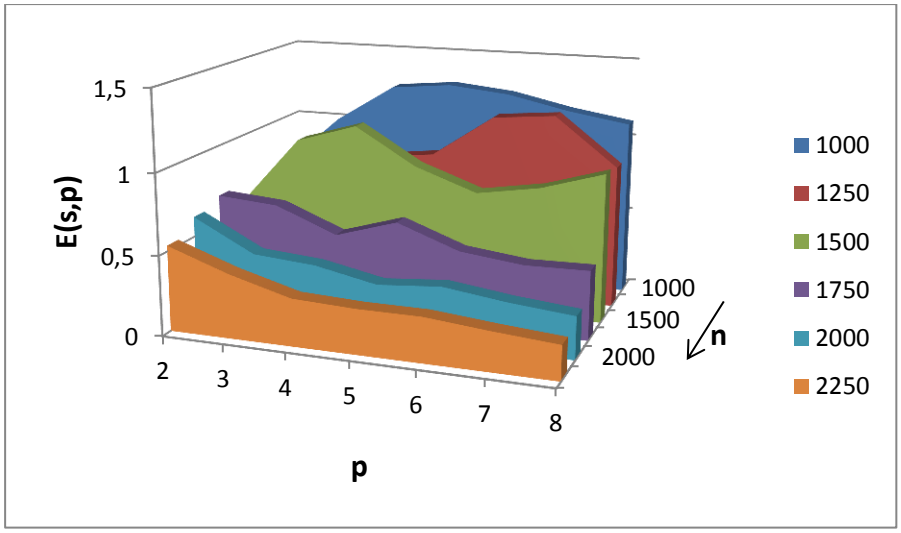
Graf na obrázku Obr. 8-19 ilustruje vplyv výpočtových a komunikačných zložiek na celkovú dobu vykonávania algoritmu, pri konštantnej záťaži (n=1000) a pri rastúcom počte procesov. Z grafu vidno, že komunikačná zložka zvyšovaním počtu procesorov rastie veľmi málo. Tento jav je v súlade s predpokladom, že v prostredí so spoločnou pamäťou je medziprocesná komunikácia veľmi rýchla a pri analýze zložitosti algoritmu počítame len s výpočtovou zložitosťou.



Obr. 8-19 Paralelné zrýchlenie

Z meraní sme odvodili jednotlivé hodnoty paralelného zrýchlenia. Hodnoty paralelného zrýchlenia sú znázornené v grafe Obr. 8-19. Paralelné zrýchlie zvyšovaním počtu procesov rastie. Pri zväčšení vstupného zaťaženia nad určitú hranicu dochádza k zníženiu hodnôt paralelného zrýchlenie. Veľkosť vstupného problému, od ktorého sa toto spomalenie rastu paralelného zrýchlenia prejaví je určená veľkosťou vyrovnávacích pamätí cache. Požiadavky na pamäťové prostriedky rastú s veľkosťou vstupného zaťaženia. V prípade daného algoritmu pamäťové požiadavky sú určované veľkosťou upravovanej matice. Najväčší vplyv na výkonnosť algoritmu má veľkosť jedného riadku matice, na ktorom sa vykonávajú výpočty. Keď požiadavky na pamäťové prostriedky presiahnu veľkosť cache, je systém nútený načítavať nové údaje priamo z globálnej operačnej pamäte. Prístup do globálnej pamäte je náročnejší a pri paralelných algoritmoch, ktoré v tom istom čase vyžadujú dáta z globálnej pamäte, dochádza k zahlteniu pamäťového systému, čo spôsobí väčšie spomalenie prístupu ku globálnej pamäti.

Na grafe Obr. 8-20 je znázornená efektivita využitia paralelných zdrojov, z ktorej taktiež vyplýva vplyv výpadku dát z cache pamäťového systému, kedy systém musí pristupovať do globálnej pamäte a efektívnosť paralelného algoritmu sa znižuje.



Obr. 8-20 Efektívnosť paralelného algoritmu

8.4 Zhrnutie vykonaných experimentov

Na základe realizovaných experimentov, získaných výsledkov a analýz môžme konštatovať, že

1. Synchrónny algoritmus GEM podľa analýzy ceny paralelného algoritmu, patrí medzi neefektívne algoritmy, ktoré nie sú vhodné pre mohutné paralelné systémy. Výsledky meraní potvrdili, že pri náraste počtu procesov sa komunikácia prudko zvyšuje a rýchlo začína dominovať medzi zložkami celkovej doby vykonania výpočtu. Komunikačné zaťaženie sa tak stáva dominantným faktorom, ktoré ohraničuje efektívnosť algoritmu. Určené hodnoty efektivity taktiež naznačujú, že tento algoritmus neefektívne využíva pridelené výpočtové zdroje. Algoritmom podobného typu, kde dochádza k častej synchronizácii nepomôžu výrazne výkonnejšie komunikačné systémy, ako skôr zníženie synchronizácie, tak aby dochádzalo k zreťazeniu výpočtov.
2. Asynchrónny algoritmus GEM podľa analýzy ceny paralelného algoritmu patrí medzi efektívne algoritmy, ktoré sú vhodné pre mohutné paralelné systémy. Výsledky meraní potvrdili, že efektivita daného algoritmu pri zvyšujúcom sa vstupnom zaťažení výrazne rastie. Pre uskutočnené meranie sa komunikácia tiež javí ako limitujúci faktor, ale je predpoklad, že pri mohutnom paralelnom systéme by došlo k úplnému zreťazeniu výpočtu a tak by bol vplyv komunikácie menší.
3. Výpočtová zložitosť hybridného algoritmu, ktorý je odvodený od asynchrónneho algoritmu, sa zvýši o dobu potrebnú na inicializáciu vlákien a implicitnú synchronizáciu. Vzhľadom na to, že asynchrónny algoritmus využíva komunikačnú funkciu broadcast optimalizovanú pre prostredie s viacjadrovými SMP počítačmi, je efektívnejší ako hybridný. Preto hybridný algoritmus je menej vhodný pre tento druh aplikácie.
4. Algoritmus výpočtu so spoločnou pamäťou funguje efektívne len do bodu, kým vstupná záťaž nevyvolá nutnosť použiť prístup do globálnej pamäte („výpadok cache“, cache miss).

Pri implementácii všetkých algoritmov boli použité optimalizované algoritmy, pre viacjadrové SMP prostredie, ktoré eliminovali problémy s redundantným odosielaním dát, ktoré bolo diskutované v časti 6.2.2.1.2.

Experimentálne výsledky potvrdili potrebu výkonnejších komunikačných systémov a efektívnejších komunikačných algoritmov, ktoré budú prispôsobené novým architektúram paralelných počítačov.

Vytvorené prediktívne modely umožňujú predikovať výkonnosti algoritmov pre ľubovoľné vstupné parametre paralelných počítačov a určiť rast efektívnosti pri zvyšujúcom sa počte procesov v závislosti na vstupnej záťaži.

9 Prínosy

9.1 Teoretické prínosy

- paralelné algoritmy
  - vývoj paralelných algoritmov a ich ďalšie smerovanie,
  - analýza dekompozičných stratégií s dôrazom na optimalizáciu výkonnosti (vplyv architektúry, paralelnej réžie - komunikácia, synchronizácia, výpočet, inicializácia),
  - aplikačné použitie štandardizovaných podporných paralelných vývojových prostredí (MPI, OpenMP)
- komunikačné mechanizmy
  - analýza komunikačných algoritmov
  - návrh optimalizácie komunikačných algoritmov pre viacjadrové SMP prostredie
- zložitosť paralelných algoritmov
  - analýza zložitosti a výkonnosti paralelných systémov (kritériá, metódy, nástroje),
  - matematické hodnotenie zložitosti algoritmov GEM
  - hodnotenie výkonnosti vyvinutých paralelných algoritmov
  - návrh komplexného experimentálneho overovania jednotlivých častí doby vykonania paralelných algoritmov (architektúra, výpočet, paralelná réžia)
- paralelné algoritmy GEM
  - návrh, vývoj
  - matematická formulácia zložitosti
  - modelovanie
  - optimalizácia
  - predikcia
- metodika experimentálnych meraní
  - návrh metodík priameho experimentálneho merania paralelných algoritmov (meranie jednotlivých zložiek celkovej doby vykonania),
  - vyhodnotenie, porovnanie a zovšeobecnenie nameraných výsledkov

9.2 Praktické prínosy

- paralelné architektúry
  - budovanie, rozširovanie a inovácia paralelných architektúr (NOW, SMP) na báze otvoreného softvéru (opensource, Linux) na Fakulte riadenia a informatiky,
  - praktické skúsenosti v rozšírení aplikačných poznatkov pri práci s paralelnými architektúrami z pohľadu vývojára a užívateľa
- zložitosť paralelných algoritmov
  - meranie a vyhodnotenie jednotlivých zložiek výkonnosti paralelných algoritmov pomocou navrhnutej metodiky priamych experimentálnych meraní
  - vyhodnotenie vplyvu architektúry paralelných systémov, paralelnej réžie (najmä komunikácie) na výkonnosť konkrétnych paralelných algoritmov
- paralelné algoritmy GEM
  - implementácia paralelných algoritmov použitím vývojových prostredí OpenMP a MPI pre architektúry
    - so spoločnou pamäťou
    - s distribuovanou pamäťou
    - hybridné(kombinácia spoločnej s distribuovanou pamäťou)

- komunikačné mechanizmy
  - Optimalizácia komunikačných algoritmov pre viacjadrové SMP prostredie
    - Implementácia broadcast komunikačného algoritmu
    - Implementácia zreťazeného broadcast komunikačného algoritmu vhodného pre asynchrónne algoritmy
- metodika experimentálnych meraní
- aplikácia navrhovaných metodík priameho experimentálneho merania paralelných algoritmov na skúmaných paralelných architektúrach

10 Záver

Aktuálnym trendom v oblasti informatiky sa stala aplikácia paralelných princípov. Od ich aplikácie sa očakáva efektívne zvyšovanie výkonnosti počítačov. Jednoznačným dôkazom daného trendu je rozmach viacjadrových procesorov, ktoré začínajú byť využívané už od mobilných zaradení cez klasické stolné počítače až po paralelné počítače (NOW,GRID). Pre efektívne využívanie paralelných systémov je potrebná komplexná analýza paralelného prostredia ako aj paralelných algoritmov. Práve pri vývoji paralelných algoritmov je nevyhnutná hlboká znalosť paralelných princípov, pretože v porovnaní so sekvenčným prostredím je paralelné prostredie komplexnejšie ovplyvnené premennými, ktoré vplývajú na celkový výkon a efektivitu systému.

Ako aplikačnú úlohu som vybral riešenie sústavy lineárnych rovníc (SLR) priamymi (finitnými) metódami. Riešenie sústavy lineárnych rovníc je vo svete modelovania fyzikálnych veličín a javov veľmi frekventovaná problematika. Od každého modelu vyžadujeme čo najlepšiu presnosť. Zvyšovaním presnosti modelu zvyšujeme aj veľkosť riešenej sústavy lineárnych rovníc až po hranicu riešiteľnosti sekvenčným spôsobom. Predložená DP analyzuje možnosti paralelizovania problematiky SLR, konkrétne so zameraním na GEM. Navrhnuté algoritmy boli podrobené analýze, pri ktorej sa určili ich teoretické vlastnosti, ktoré boli experimentálne overené. Navrhnuté algoritmy boli optimalizované pre toho času čoraz viac používané paralelné prostredia pozostávajúce z viacjadrových SMP počítačov prepojených do GRIDu, NOW atď. Optimalizácia spočíva v implementovaní efektívnych komunikačných metód, ktoré využívajú vlastnosti daného prostredia.

Pre aplikačnú oblasť je potrebná podrobná analýza paralelného prostredia, algoritmov, komunikačných nárokov a princípov paralelného počítania. Čo bolo cieľom danej práce, ktorá chce prispieť k lepšej optimalizácii a výkonnosti distribuovaných systémov.

11 Literatúra

[1] ABDERAZEK, A.B. *Multicore systems on-chip -- Practical Software/Hardware design*. 2nd Editio. vyd. [s.l.]: Imperial college press, 2013. .

[2] AKHTER, S. - ROBERTS, J. *Multi-Core Programming*. USA: Intel Press, 2006. ISBN 0-9764832-4-6.

[3] ANDREWS, G.R. *Foundations of Multithreaded, Parallel and Distributed Programming*. [s.l.]: Addison Wesley, 2000. .

[4] ANTHONY, M. - HARVEY, M. *Linear Algebra: Concepts and Methods* [online]. [s.l.]: Cambridge University Press, 2012. ISBN 9780521279482.

[5] BAASE, S. - GELDER, A. VAN *Computer Algorithms -- Introduction to Design and Analysis*. USA: Addison Wesley, 2000. .

[6] BARRY, W. *Parallel Programming: Techniques And Applications Using Networked Workstations And Parallel Computers, 2/E* [online]. [s.l.]: Pearson Education, 2006. ISBN 9788131702390.

[7] BERMAN, F. et al. *Grid Computing: Making the Global Infrastructure a Reality*. [s.l.]: John Wiley&Sons, 2003. .

[8] BISCHOF, C. *Parallel computing: architectures, algorithms, and applications*. [s.l.]: IOS Press, 2008. .

[9] BLAIR-CHAPPELL, S. - STOKES, A. *Parallel Programming with Intel Parallel Studio XE* [online]. [s.l.]: Wiley, 2012. ISBN 9781118221136.
[10] BURNS, M. V et al. Modeling and simulative perf. analysis of SMP and clusters,. In *Simulation*. 2000. Vol. 74, s. 74–92. .
[11] CASANOVA, H. et al. *Paralel Algorithms*. [s.l.]: CRC Press, 2008. .
[12] CODENOTI, B. - LEONCINI, M. *Parallel komplexity of linear systems*. United Kingdom: Imperial college press, 1991. .
[13] COOPER, K. et al. *Languages and Compilers for Parallel Computing: 23rd International Workshop, LCPC 2010, Houston, TX, USA, October 7-9, 2010. Revised Selected Papers* [online]. [s.l.]: Springer, 2011. ISBN 9783642195945.
[14] CORMEN, H.T. et al. *Introduction to Algorithms*. [s.l.]: MIT Press and McGraw-Hill, 2009.
[15] COULOURIS, G. et al. *Distributed Systems -- Concepts and Design (5th Edition)*. 5. vyd. [s.l.]: Addison Wesley, 2011. .
[16] DASGUPTA, S. et al. *Algorithms*. [s.l.]: McGraw-Hill, 2006. .
[17] DESPREZ, F. *Algorithms And Tools for Parallel Computing on Heterogeneous Clusters* [online]. [s.l.]: Nova Science Publishers, Incorporated, 2007. ISBN 9781600210495.
[18] DIMOPOULOS, N.J. - LI, K.F. *High Performance Computing Systems and Applications*. [s.l.]: Springer, 2001. .
[19] DUMMLER, J. et al. *Combined Scheduling and Mapping for Scalable Computing with Parallel*. [s.l.]: IOS Press, 2012. .
[20] EDMONDS J. *How to think about algorithms*. [s.l.]: Cambridge University Press, 2010. .
[21] EECKHOUT, L. *Computer Architecture Performance Evaluation Methods*. [s.l.]: Morgan & Claypool Publishers, 2010. .
[22] EL-REWINI, H. - ABD-EL-BARR, M. *ADVANCED COMPUTER ARCHITECTURE AND PARALLEL PROCESSING*. New Jersey, Hoboken: John Wiley & Sons, 2005. ISBN 0-471-46740-5.
[23] FAYEZ, G. *Computer and Communication Networks* [online]. [s.l.]: Springer Science+Business Media, 2008. ISBN 9780387744377.
[24] FORTIER, P. - HOWARD, M. *Computer system performance evaluation and prediction*. [s.l.]: Digital Press, 2003. .
[25] FORUM, M.P.I. MPI: a Message-Passing Interface standard. In. 2013. .
[26] FOSTER, I. et al. *Cloud Computing and Grid Computing 360-Degree Compared*. Austin, TX: Dept. of Comput. Sci., Univ. of Chicago, Chicago, IL, USA, 2009. .
[27] FOSTER, I. - KESSELMAN, C. *The Grid 2 - Blueprint for a New Computing Infrastructure (Second Edition)*. USA: Morgan Kaufmann, 2003. .
[28] FRACHTENBERG, E. - SCHWIEGELSHOHN, U. *Job Scheduling Strategies for Parallel Processing: 15th International Workshop, JSSPP 2010, Atlanta, GA, USA, April 23, 2010, Revised Selected Papers* [online]. [s.l.]: Springer, 2010. ISBN 9783642165047.
[29] GEBALI, F. *Algorithms and Parallel Computing* [online]. [s.l.]: Wiley, 2011. ISBN 9780470934630.
[30] GOLDREICH, O. *Computational complexity*. [s.l.]: Cambridge University Press, 2010. .
[31] GOLUB, G.H. - LOAN, C.F. VAN *Matrix Computations* [online]. [s.l.]: Johns Hopkins University Press, 2012. ISBN 9781421407944.
[32] GRAMA, A. et al. *Introduction to Parallel Computing, Second Edition*. 2. vyd. [s.l.]: Addison Wesley, 2003. ISBN 0-201-64865-2.
[33] GRAMA, A. et al. *Measuring the Scalability of Parallel Algorithms and Architecture*. University of Minnesot: The MIT press, 1993. .
[34] HANULIAK, I. *Parallel architecrutes - multiprocessors, computer networks(in Slovak)*. Žilina: Book center, 1997. .
[35] HANULIAK, I. *Parallel computers and algorithms (in Slovak)*. Slovakia: Publ.: ELFA Košice, 1999. .
[36] HANULIAK, J. *Modelovanie a predikcia výkonnosti distr. par. algoritmov - DP*. Žilina: Žilinská univerzita, Fakulta riadenia a informatiky, 2006. .
[37] HANULIAK, P. Comparison of HPC and GRID systems. In *Veda 2006*. 2006. s. 140–146. .
[38] HANULIAK, P. Virtual parallel computer. In *TRANSCOM 2007*. Žilina: TRANSCOM 2007, 2007. .

[39] HANULIAK, P. - HANULIAK, I. Performance evaluation of iterative parallel algorithms. In *Kybernetes*. 2007. Vol. 36. .
[40] HELD, J. et al. *From a Few Cores to Many: A Tera-scale Computing Research Overview, White Paper*. 2006. .
[41] HEURING, V.P. - JORDAN, H.I. *Computer Systém Design and Architecture*. New Jersey: Pearson Prentice Hall, 2004. .
[42] HOFFMANN, K.H. - MEYER, A. *Parallel algorithms and cluster computing: implementations, algorithms and applications*. [s.l.]: Springer, 2006. .
[43] HOLÚBEK, A. *Performance Prediction and Optimization of Parallel Algorithm*. Poznan: InterTech 2010, 2010. .
[44] HOLÚBEK, A. *Performance prediction of parallel systems*. Kunovice: ICSC 2010, 2010. .
[45] HOWARD, A. *Elementary Linear Algebra*. [s.l.]: John Wiley and Sons, 2010. .
[46] HSIUNG, P.A. et al. *Multicore Hardware-Software Design and Verification Techniques* [online]. [s.l.]: Bentham Science Publishers, 2011. ISBN 9781608052257.
[47] HUGHES, C. - HUGHES, T. *Parallel and Distributed Programming Using C++*. [s.l.]: Addison Wesley, 2004. .
[48] CHANDRA, R. et al. *Parallel Programming in OpenMP*. [s.l.]: Morgan Kaufmann Publishers, 2000. .
[49] CHAPMAN, B.J.G. - PAS, R. Van der *Using OpenMP - Portable Shared Memory Parallel Programming*. Massachusetts, USA: The MIT Press, 2008. .
[50] J., B.A. *Communication network and computer systems*. [s.l.]: Imperial College Press, 2004. .
[51] JANOVIČ F., H.P. *Optimisation of decomposition strategy in parallel matrix algorithms*. Žilina: In Proc. ICTIC, 2012. .
[52] JEFFERS, J. - REINDERS, J. *Intel Xeon Phi Coprocessor High Performance Programming* [online]. [s.l.]: Elsevier Science & Technology Books, 2013. ISBN 9780124104143.
[53] JODY, I. *Linpack* [online]. [s.l.]: Cred Press, 2011. ISBN 9786136611976.
[54] JOSEPH, J. - FELLENSTEIN, C. *Grid computing*. [s.l.]: Prentice Hall PTR, 2004. .
[55] KAMINOV, I.P. et al. *Distributed and parallel systems -- Cluster and Grid computing*. [s.l.]: Springer-Verlag, 2005. .
[56] KEPNER, J. V *Parallel MATLAB for multicore and multinode computers* [online]. [s.l.]: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009. ISBN 9780898718126.
[57] KORNAROS, G. *Multi-Core Embedded Systems* [online]. [s.l.]: Taylor & Francis, 2010. ISBN 9781439811627.
[58] LASTOVETSKY, A. et al. *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 15th European PVM/MPI Users' Group Meeting, Dublin, Ireland, September 7-10, 2008, Proceedings* [online]. [s.l.]: Springer, 2008. ISBN 9783540874744.
[59] MAGOULES, F. et al. *Grid Ressource Management: Towards Virtual and services Complaint grid Computing*. [s.l.]: CRC Press, 2008. .
[60] MAGOULES, J. et al. *Introduction to Grid Computing*. [s.l.]: CRC Press, 2008. .
[61] MAOZHEN, L. - BAKER, M. *The grid: core technologies*. [s.l.]: John Wiley and Sons, 2005.
[62] MATIAŠKO, K. - HANULIAK, J. To complexity evaluation of parallel algorithms. In *5th Int. Conf. "'NIT"'*. 2002. s. 28–34. .
[63] MILLER, R. - BOXER, L. *Algorithms -- Sequential and Parallel*. [s.l.]: Prentice Hall, 2000. .
[64] MINOLI, D. *A networking approach to grid computing*. [s.l.]: John Wiley and Sons, 2005.
[65] MORET, B.M.E. - SHAPIRO, H.D. *Algorithms from P to NP: Design & efficiency* [online]. [s.l.]: Benjamin/Cummings, 1991. ISBN 9780805380088.
[66] MURTHY, C.S.R. et al. *New parallel algorithms for direct solution of linear equations* [online]. [s.l.]: J. Wiley, 2001. ISBN 9780471361657.
[67] NAIKSATAM, S. - UNIVERSITY, S.C. *Supernetworking the Metacomputer: Enabling Guaranteed Bandwidth Through Deterministic and Efficient Provisioning* [online]. [s.l.]: Santa Clara University, 2006. ISBN 9780542944451.
[68] OBAIDAT, M.S. - BOUDRIGA, N.A. *Fundamentals of Performance Evaluation of Computer and Telecommunications Systems*. [s.l.]: John Wiley & Sons, 2010. .

[69] PADUA, D. *Encyclopedia of Parallel Computing* [online]. [s.l.]: Springer, 2011. ISBN 9780387097657.
[70] PACHECO, P. *An Introduction to parallel computing*. [s.l.]: Morgan Kaufmann, 2011. .
[71] PATTERSON, D.A. - HENNESSY, J.L. *Computer Organization and Design, Fourth Edition*. [s.l.]: Morgan Kaufmann, 2008. .
[72] PETERSON, L.L. - DAVIE, B.C. *Computer networks - a system approach*. [s.l.]: Morgan Kaufmann, 2011. .
[73] PREVE, N.P. *Grid computing* [online]. [s.l.]: Springer London, Limited, 2011. ISBN 9780857296764.
[74] QUINN, M.J. *Parallel Programming in C with MPI and Open MP (First Edition)*. [s.l.]: McGraw-Hill Publishers, 2004. .
[75] QUINN, M.J. *Parallel programming in C with MPI and OpenMP*. [s.l.]: McGraw-Hill Higher Education, 2004. .
[76] RADHAKRISHNAN, R. *Computer Organization And Architecture* [online]. [s.l.]: Prentice-Hall Of India Pvt. Limited, 2007. ISBN 9788120332003.
[77] RAUBER, T. - RUNGER, G. *Parallel Computing: From Multicore and GPU's to Petascale*. [s.l.]: IOS Press, 2010. .
[78] RAUBER, T. - RÜNGER, G. *Parallel Programming for Multicore and Cluster Systems*. [s.l.]: Springer Verlag, 2010. .
[79] RAUCHWERGER, L. *Languages and Compilers for Parallel Computing*. USA: Springer Science, 2005. .
[80] REINDERS, J. *Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism* [online]. [s.l.]: O'Reilly Media, 2010. ISBN 9781449390860.
[81] RESCH, R. - GABRIEL, E. *Supercomputers in Grids*. [s.l.]: Int. Journal of Grid and HPC, 2009. .
[82] RUSSELL, J. - COHN, R. *Mpich* [online]. [s.l.]: Book on Demand, 2012. ISBN 9785510869743.
[83] RUSSELL, J. - COHN, R. *Openmp* [online]. [s.l.]: Book on Demand, 2012. ISBN 9785512427415.
[84] SANJEEV, A. - BOAZ, B. *Computational Complexity, 1th edition*. United Kingdom: Cambridge University Press, 2009. .
[85] SHAN, H. Message passing and shared address space parallelism on an SMP cluster. In *Parallel computing*. 2007. Vol. 29, s. 167–186. .
[86] SHIRAZI, B.A. et al. *Scheduling and local balancing in Parallel and Distributed Systems*. [s.l.]: John Wiley&Sons, 1995. .
[87] SOUDRIS, D. - JANTSCH, A. *Scalable Multi-Core Architectures: Design Methodologies and Tools* [online]. [s.l.]: Springer New York, 2012. ISBN 9781441967787.
[88] STALLINGS, W. *Computer Organisation and Architecture -- Designing for Performance (Fifth Edition)*. [s.l.]: Prentice Hall, 2003. .
[89] STANEEVSKA, S.K. et al. *Grid and Cloud Computing* [online]. [s.l.]: Springer Berlin Heidelberg, 2010. ISBN 9783642052149.
[90] VLASSOV, V. - AYANI, R. Analytical modeling of multithreaded architectures. In *Journal of system architecture*. 2000. Vol. 46, s. 1205–1230. .
[91] VU, Q.H. et al. *Peer-to-Peer Computing* [online]. [s.l.]: Springer Berlin Heidelberg, 2010. ISBN 9783642035319.

Publikačná činnosť autora

V zahraničí

Use of parellelism in the field of speech recognition / Hyben Martin, Húdik Martin.
In: Process control 2012 [elektronický zdroj] : 10th international conference : Kouty nad Desnou, June 11-14, 2012, Czech Republic. - Pardubice: University of Pardubice, 2012. - ISBN 978-80-7395-500-7. - CD-ROM, C042a [5] s.
Poznámka:
Vyšiel aj tlačený zborník abstraktov; S. 26

Load balancing in grid with use of agents / Húdik Martin, Púchyová Jana.
In: Process control 2012 [elektronický zdroj] : 10th international conference : Kouty nad Desnou, June 11-14, 2012, Czech Republic. - Pardubice: University of Pardubice, 2012. - ISBN 978-80-7395-500-7. - CD-ROM, C025a [4] s.
Poznámka:
Vyšiel aj tlačený zborník abstraktov; S. 31

Performance optimization of broadcast collective operation on multi-core cluster [Optimalizovanie výkonosti kolektívnych komunikačných operácii v multi-core prostredí] / Hudik Martin.
In: ICSC 2012 : tenth international conference on soft computing applied in computer and economic environments : January 20, Kunovice, Česká republika : proceedings. - Kunovice: European Polytechnic Institute, 2012. - ISBN 978-80-7314-279-7. - S. 51-55.

Doma

Analyzing performance of parallel algorithms for linear system equations [Analýza výkonu paralelných algoritmov pre výpočet sústavy lineárnych rovníc] / Hanuliak Peter, Hudik Martin.
In: GCCP 2011 : 7th international workshop on grid computing for complex problems : October 24-26,2011, Bratislava, Slovakia : proceedings. - [Bratislava: Institute of informatics SAS, 2011]. - ISBN 978-80-970145-5-1. - [8] s.

Parallel complexity of linear system equations / Húdik Martin, Hanuliak Peter.
In: TRANSCOM 2011 : 9-th European conference of young research and scientific workers : Žilina, June 27-29, 2011, Slovak Republic. - Žilina: University of Žilina, 2011. - ISBN 978-80-554-0372-4. - S. 107-110.

Communication costs in parallel machines [Komunikačné náklady v paralelných počítačoch] / Martin Húdik.
In: Zimná škola MICT [elektronický zdroj] : mathematics for information and communication technologies : 6th winter school of mathematics for ICT : Šachtičky 3.-8.1.2011. - Banská Bystrica: Science and Research Institute, MBU, 2011. - ISBN 978-80-557-0252-0. - S. 58-60.