VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

TAKTOVÁNÍ MODERNÍCH PROCESORŮ S OHLEDEM NA VÝKON, SPOTŘEBU A TEPLOTU

OVERCLOCKING OF MODERN PROCESSORS WITH AN EMPHASIS ON PERFORMANCE, POWER CONSUMPTION AND TEMPERATURE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAKUB KELEČÉNI
AUTHOR

VEDOUCÍ PRÁCE Ing. VOJTĚCH NIKL
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Kelečéni Jakub**
Obor: Informační technologie
Téma: **Taktování moderních procesorů s ohledem na výkon, spotřebu a teplotu**
Overclocking of Modern Processors with an Emphasis on Performance, Power Consumption and Temperature
Kategorie: Počítačová architektura

Pokyny:

1. Seznamte se s architekturou dnešních procesorů Intel a s možností změny jejich frekvence a napětí s ohledem na výkon, spotřebu a provozní teplotu.
2. Navrhněte sadu benchmarků, které otestují vhodnost dané konfigurace procesorů pro daný problém s ohledem na celkovou dobu výpočtu, efektivitu a s tím spojenými provozními náklady.
3. Benchmarky implementujte.
4. Otestujte implementované kódy na sadě dostupných procesorů.
5. Zhodnoťte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvního a druhého bodu zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Nikl Vojtěch, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Táto práca rieši problematiku závislosti – celkovej doby výpočtu, spotreby energie a teploty – na pracovnej frekvencií serverového procesora. V teoretickej časti je popísaná architektúra použitého procesora, sada benchmarkov a druhy algoritmov. Praktická časť je zameraná na testovanie navrhnutej sady benchmarkov (násobenie matíc, quicksort, výpočet π, Ackermannova funkcia, LAMMPS, PMBW, Linpack). Sada benchmarkov pozostáva z jednovláknových a paralelných algoritmov. Testovanie prebiehalo pri nastavení troch rôznych frekvencií CPU a pri spustení paralelných benchmarkov na rôznom počte výpočtových vlákien. Pri každom teste boli zaznamenávaná údaje o spotrebe CPU a RAM. V práci je zohľadnený vplyv paralelizácie na spotrebu energie a na čas výpočtu. Získané údaje sú zhrnuté do tabuliek a grafov. Výsledkom práce je zhodnotenie vhodnosti konfigurácie CPU s ohľadom na čas výpočtu a spotrebu energie, pre jednotlivé benchmarky. Zo získaných výsledkov vyplýva, že vhodnosť použitej frekvencie CPU je závislá od charakteru výpočtového problému, a tiež od požiadavky pre dosiahnutie najlepšieho času, alebo spotreby.

Abstract

This thesis analyzes the dependency of performance, power consumption and temperature on processor frequency. Theoretical part discusses the processor architecture, benchmarks and algorithm types. Experimental part is focused on benchmarks–matrix multiplication, Quicksort, π number calculation, Ackermann function, LAMMPS, PMBW, Linpack. This set of benchmarks includes both single-threaded and multi-threaded algorithms. Testing consist of three different settings of processor frequency. Multi-threaded benchmarks using different number of threads. Informations regarding the power consumption of CPU and RAM were recorded during these tests. Every test logs his running time. The impact of parallelization on power consumption and runtime is also reflected. Results from the tests are shown in charts and tables. The proper configuration of CPU for each given algorithm is analyzed in conclusion.

Kľúčové slová

Pretaktovanie, Spotreba, Teplota, Procesor, Intel Xeon, PAPI, MPI, OpenMP, Násobenie matíc, Quicksort, PMBW, LAMMPS, FLOPS.

Keywords

Overclock, Power consumption, Temperature, Processor, Intel Xeon, PAPI, MPI, OpenMP, Matrix multiply, Quicksort, PMBW, LAMMPS, FLOPS.

Citácia

KELEČÉNI, Jakub. *Taktování moderních procesorů s ohledem na výkon, spotřebu a teplotu.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Nikl Vojtěch.

Taktování moderních procesorů s ohledem na výkon, spotřebu a teplotu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Vojtecha Nikla.

Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

. . . . . .
Jakub Kelečéni
17. mája 2016

Obsah

Kapitola 1

Úvod

Od uvedenia prvého počítača, ktorého súčasťou bol mikroprocesor (CPU) bolo snahou dosiahnuť čo najvyššiu rýchlosť tohto komponentu. V priebehu času bolo vyvinutých niekoľko typov architektúr a rôznych technológií. V minulosti bolo dosiahnutie vyššieho výkonu CPU zabezpečené zvyšovaním pracovnej frekvencie. Časom sa ukázalo, že frekvenciu nie je možné zvyšovať do „nekonečna“, takže bolo potrebné nájsť iné riešenia. Medzi tieto riešenia patrí zvyšovanie počtu jadier CPU a pridanie virtuálnych vlákien (Hyper Threading).

Vyšší kmitočet pracovnej frekvencie CPU nám prináša lepší výkon. Znamená to, že CPU dokáže spracovať viac operácií za jednotku času. Avšak pretaktovanie spôsobuje aj negatívne dôsledky. Je to napríklad zvyšujúca sa spotreba a tiež množstvo vyprodukovaného tepla. V praxi to znamená potrebu výkonnejšieho chladiaceho systému a drahšiu prevádzku počítača.

Tieto dôsledky nás nútia zamyslieť sa nad tým, ako vplýva pracovná frekvencia na dobu výpočtu, spotrebu energie a teplotu. A tiež zhodnotiť, že či nie je výhodnejšie použiť nižšiu frekvenciu za účelom nižšej spotreby, avšak vyšším časom potrebným na výpočet.

Cieľom tejto práce je zoznámenie sa s architektúrou dnešných procesorov Intel, možnosťou zmeny pracovnej frekvencie a jej vplyvu na spotrebu, teplotu a čas výpočtu. Testovanie tohoto vplyvu bude prebiehať na sade benchmarkov, ktorú je potrebné navrhnúť.

Výsledkom praktickej časti je testovanie niekoľkých algoritmov. Po analýze výsledkov testov bude možné zhodnotiť, že pre aké typy výpočtových problémov je vhodnejšie procesor podtaktovať, z ohľadom na spotrebu a dobu výpočtu, a naopak kedy je vhodnejšie použiť čo najvyššiu frekvenciu. Práca sa bude zaoberať aj vplyvom frekvencie CPU na rýchlosť pamäte, a tiež na testovanie reálnej aplikácie. Sada benchmarkov zahŕňa jednovláknové aj viacvláknové algoritmy.

Kapitola 2

Procesory Intel

Spoločnosť Intel je najstaršou a jednou z najväčších spoločností, zaoberajúcich sa vývojom mikroprocesorov. Produkty tejto spoločnosti nachádzajú uplatnenie v mnohých odvetviach (elektronika, automobilový priemysel, automatizácie, medicína, energetika, atď.) [6]. Vzhľadom k veľkému podielu produktov na trhu, má význam zamerať sa v tejto práci na procesory od tohoto výrobcu.

2.1 Procesor

Procesor označovaný ako CPU – Central processing unit je základnou elektronickou súčasťou počítača. Vykonáva strojové inštrukcie, z ktorých je zložený počítačový program a obsluhuje vstupy a výstupy nad týmto programom. V dnešnej dobe existujú rôzne typy procesorov a architektúr, viď [20].

Tento komponent má najväčší vplyv na rýchlosť celého počítačového systému. Kľúčovú úlohu zohráva aj pri celkovej spotrebe počítača. Z tohoto dôvodu má význam riešiť problém frekvencie a spotreby procesoru.

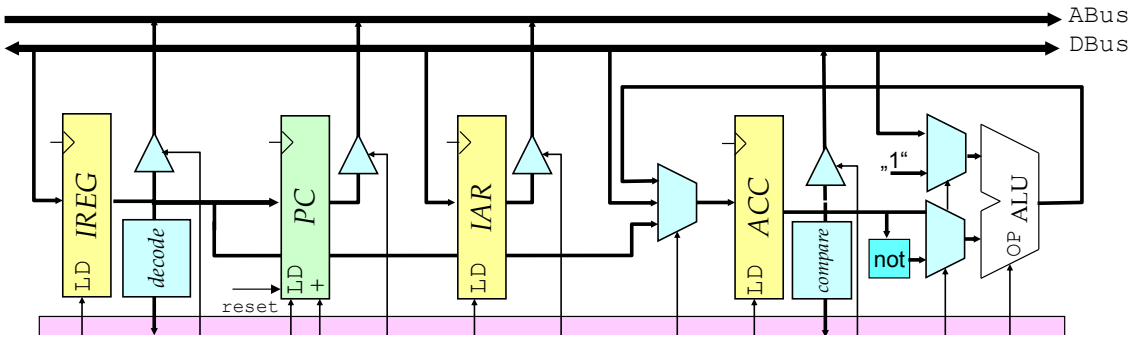
Z konštrukčného hľadiska je základom procesoru kremíková doska, na ktorej je umiestnený logický obvod. Úlohou obvodu je spracovanie sady jednoduchých príkazov. Z pohľadu programátora znamená jeden príkaz určitú postupnosť jednotiek a núl. Program v takomto tvare je označovaný ako strojový kód. Programovanie v strojovom kóde je v súčasnosti nepredstaviteľné, preto bol vytvorený jazyk symbolických inštrukcií – Asembler. Asembler je spoločný pre jednu architektúru, čo znamená, že v rôznych architektúrach sa inštrukčná sada môže odlišovať.

Z hľadiska synchronizácie sa procesor zaraďuje medzi synchrónne zariadenia. Využíva sa hodinový (synchronizačný) signál generovaný kryštálom. Od frekvencie signálov sa odvíja množstvo operácií vykonateľných procesorom za jednotku času.

Procesor sa skladá s niekoľkých hlavných komponentov (viď Obr. 2.1):

- Programový čítač (PC) – určuje adresu, na ktorej sa nachádza nasledujúca inštrukcia (pri inicializácií implicitná hodnota).
- Inštrukčný register (IR) – uchováva práve spracovávanú inštrukciu.
- Radič – riadi činnosť CPU.
- Aritmeticko-logická jednotka (ALU) – slúži k vykonávaniu matematických operácií.

- I/O jednotka – umožňuje vstup/výstup dát z/do CPU do registrov aj iným spôsobom ako prostredníctvom pamäte.



Obr. 2.1: Bloková schéma jednoduchého procesora (prevzaté z [28]).

Tieto komponenty zabezpečujú činnosť CPU, ktorá je nasledovná: procesor vykonáva program, ktorý je uložený od zadanej adresy v pamäti. Program transformuje dáta uložené na zadanej adrese v pamäti, na iné dáta, ktorých umiestnenie musí byť presne špecifikované. V prípade, že CPU obsahuje I/O jednotku, tak môže zároveň pri spracovaní využívať informácie so svojich vstupných portov a produkovať dáta na výstupné porty [28].

Okrem spomínaných komponentov, dnešné procesory obsahujú rôzne špeciálne registre uchovávajúce operandy a medzivýsledky, matematický koprocesor (FPU), vektorovú jednotku (vektory desatinných čísiel), atď. Dôležitou súčasťou je aj rýchla vyrovnávacia pamäť cache, ktorá uchováva najčastejšie používané dáta a príkazy (inštrukcie). Typicky pozostáva z viacerých úrovní – L1 nachádza sa v jadre (najrýchlejšia, do 256 kB), L2 (do 12 MB), L3 zdieľaná (najpomalšia, do 256 MB). V procesoroch je tiež implementované prúdové spracovanie inštrukcií (pipelining).

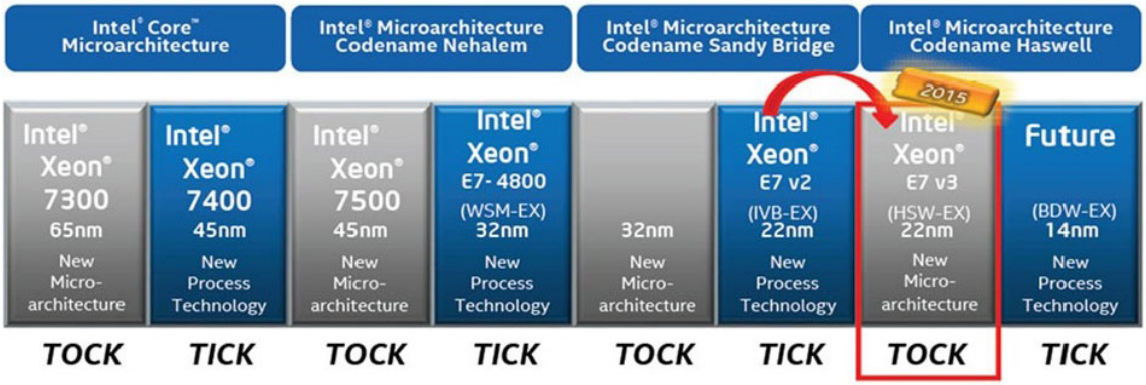
2.2 Architektúra Intel Xeon

Táto práca je zameraná na architektúru procesorov Intel Xeon. Ide o typ architektúry, ktorá nie je určená pre osobné počítače a notebooky. Zameriava sa na pracovné stanice a servery, ktoré vyžadujú vysoký výkon. Od komerčných procesorov sa odlišujú v niekoľkých faktoroch. Napríklad v podpore EEC (Error-correcting code) pamätí, optimalizácií pre chod 24/7, maximálnej veľkosti pamäte, podpora multisocket, atď [13]. Vývoj tejto architektúry ilustruje Obr. 2.2.

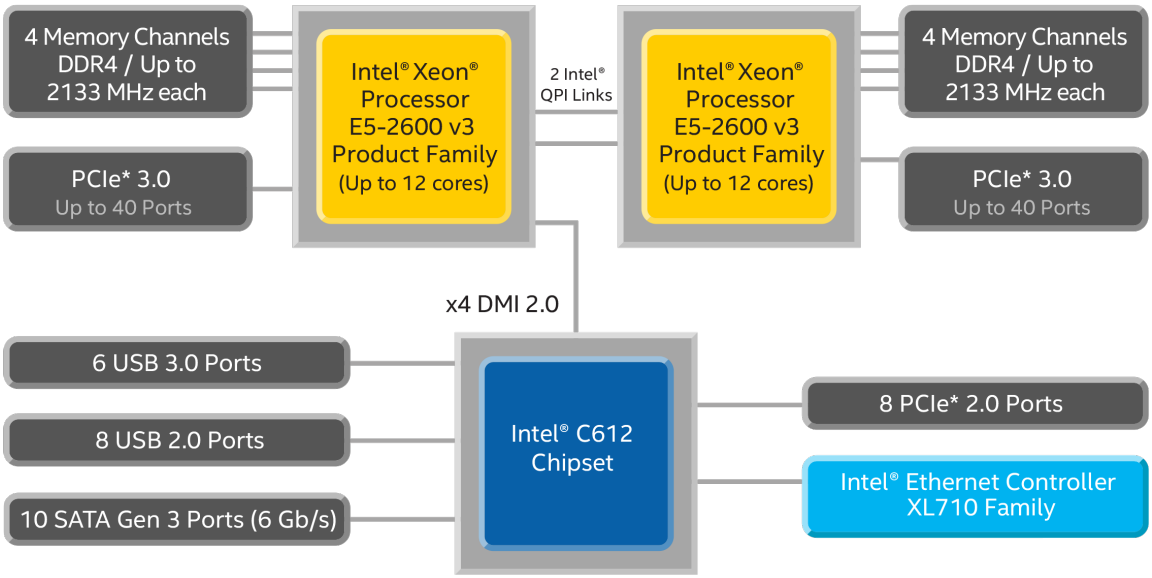
Súčasná generácia procesorov Xeon E5 je založená na architektúre `Haswell` a je prvou generáciou CPU, ktorá prináša podporu až 12 fyzických jadier na jednom CPU. Taktiež podporuje konfiguráciu dvoch pätíc, čo znamená až 24 fyzických jadier [9]. Architektúru popisuje bloková schéma viď Obr. 2.3.

Medzi kľúčové vlastnosti tejto architektúry patrí [22]:

- 22 verzií CPU – 4 až 18 jadier
- základné frekvencie v rozsahu 1,6 GHz – 3,5 GHz
- podpora DDR4 pamätí
- 40 GbE LAN



Obr. 2.2: Intel Xeon Tick Tock model (prevzaté z [24]).



Obr. 2.3: Intel® Xeon® Processor E5-2600 v3 Product Family with Intel® C612 Chipset: Diagram (prevzaté z [10]).

- podpora AVX 2.0 - rozšírenie inštrukcií SSE a AVX na 256 bitov
- 2,5 MB vyrovnávacej pamäte pripadajúcej na jadro

V porovnaní s bežnými procesormi Intel Core sa architektúra Xeon podstatne odlišuje aj napriek tomu, že majú mnoho spoločného. Hlavný rozdiel je v podpore technológií `uncore`, vďaka ktorým je možné zostaviť veľmi rýchle systémové bloky. Práve z tohoto dôvodu sa procesory Xeon nachádzajú v najvýkonnejších superpočítačoch sveta. Taktiež sú súčasťou mnohých pracovných staníc, ktoré sú využívané v oblasti výskumu.

Technológie uncore v týchto procesoroch v porovnaní s prechádzajúcimi generáciami zabezpečujú vyššiu rýchlosť prenosu a spracovania dát procesorom. Tieto vylepšenia sú dosahované vďaka nasledovným technológiám [3]:

- Intel® Direct I/O technology – 2,3 násobná akcelerácia vstupno-výstupných operácií

v porovnaní s predchádzajúcou generáciou.

- Intel Integrated I/O – dvojnásobné zvýšenie priepustnosti zbernice, ide o prvý procesor typu Xeon v ktorého čipe je integrovaná zbernica PCIe 3. generácie.
- Pridané dva pamäťové kanály – rýchlejší a efektívnejší prístup k dátam v pamäti.
- Intel® vPro technology – lepší manažment užívateľov.

2.2.1 Parametre dostupného CPU

Procesor použitý pre testovanie v rámci tejto práce, je typu Intel® Xeon® CPU E5-2620 v3. Obsahuje ho sprístupnený školský server, ktorého bližšia špecifikácia je popísaná v Kap. 5.1. Parametre každého procesoru Intel, sú uvedené na webových stránkach výrobcu. Vybrané parametre použitého procesora [5]:

- *Intel Smart Cache – 15 MB*
- *Intel QPI speed – 8 GT/s*
- *Lithography – 22 nm*
- *VID Voltage Range – 0.65 V–1.30 V*
- *# of Cores – 6*
- *Processor Base Frequency – 2.4 GHz*
- *Max Turbo Frequency – 3.2 GHz*
- *TDP – 85 W*
- *Max Memory Size – 768 GB*
- *Max Memory Bandwidth – 59 GB/s*

Veľkosť rýchlej vyrovnávacej pamäte tohoto procesora je nasledovná: L1 6×32 kB instruction cache a 6×32 kB data cache, L2 6×256 kB, L3 15 MB. Použitý procesor obsahuje 6 jadier. Každé procesorové jadro má vlastnú pamät úrovne L1 a L2. Dostupná pamäť úrovne L3 je zdieľaná pre celý procesor.

2.3 Zmena frekvencie

Taktovanie procesorov znamená zvyšovanie frekvencie za účelom dosiahnutia vyššieho výkonu – vyšší takt je priamo úmerný počtu operácií, ktoré dokáže CPU spracovať za jednotku času. Zmena frekvencie je možná niekoľkými spôsobmi. Jedným z nich je úprava frekvencie procesora cez BIOS. Tento prístup môže byť za určitých okolností problematický, keďže do nastavení BIOSu je možné pristúpiť iba pri štarte počítača. Zmenu parametrov ovplyvňujúcich frekvenciu procesora navyše nemusí podporovať každá základná doska a každý procesor. Pri nesprávnom nastavení, alebo málo výkonnom chladení, môže dôjsť k poškodeniu CPU alebo niektorého s ostatných komponentov. Problémom pri zmene frekvencie takýmto spôsobom je tiež strata záruky poskytovanej výrobcom.

Moderné procesory Intel sú schopné meniť svoju frekvenciu dynamicky za behu operačného systému na základe aktuálneho zaťaženia (Intel Turbo, Intel SpeedStep). Vďaka tomu je možné meniť frekvenciu podľa požiadaviek užívateľa, či už voľbou režimu (vysoký výkon, šetrenie energie, atď.) alebo podľa požiadavky na konkrétnu frekvenciu. Tieto nastavenia sú

však obmedzené v rozsahu maximálnej a minimálnej pracovnej frekvencie procesora. V konečnom dôsledku ide skôr o podtaktovanie, alebo nastavenie špecifickej frekvencie. V rámci tejto práce budeme používať práve tento prístup ku zmene frekvencie procesora.

Od frekvencie procesora sa odvíja jeho výkon. So zvýšeným výkonom sa menia aj iné parametre, ktoré nemusia byť žiadúce. Jedným z problémov, ktorý vzniká pri pretaktovaní procesora, je väčšie množstvo odpadového tepla. V prípade osobného počítača je tento problém možné riešiť použitím výkonnejšieho chladiaceho systému. Avšak ak budeme uvažovať o výkonnejších a komplexnejších systémoch, akými sú výkonné servery a superpočítače, toto riešenie nebude realizovateľné. Pri takýchto systémoch je potrebné brať zreteľ aj na spotrebovanú energiu. Predpokladajme, že vysoká frekvencia nemusí v každom prípade znamenať akceleráciu výkonu. Ak by sa čas výpočtu problému na nižšej frekvencií rovnal alebo približoval tomu na vyššej, prídeme k záveru, že sme schopný znížiť náklady na prevádzku takéhoto systému vďaka menšieho množstva spotrebovanej energie procesorom a chladiacim systémom. Táto úvaha je jednou z motivácií vzniku tejto práce.

2.4 Spotreba

Spotreba procesora je priamo úmerná jeho vyťaženiu. Závislá je tiež na jeho pracovnej frekvencií. Intel uvádza pri každom CPU hodnotu TDP. TDP (Thermal Design Power) predstavuje maximálny teplotný výkon, ktorý môže procesor rozptýliť počas behu na plnú záťaž, pričom výrobca garantuje spoľahlivosť prevádzky. Za normálnych okolností celková spotreba procesora nemôže presiahnuť hodnotu TDP. Každý procesor má túto hodnotu špecifickú a je uvedená v technických údajoch procesora. Pri procesoroch typu Xeon sa táto hodnota pohybuje v rozsahu 40 W – 130 W.

Pre meranie spotreby je možné využiť viacero možností. Najjednoduchšou možnosťou je použitie špeciálneho prístroja – wattmetra, ktorý meria spotrebovaný výkon. Wattmeter sa pripojí na napájanie CPU a zobrazuje spotrebu v reálnom čase. Pre naše účely tento spôsob nie je vhodný, pretože budeme pracovať na vzdialenom servery. Problémom môže byť aj to, že je vyžadovaný priamy prístup k hardvéru, čo zo sebou nesie určité riziká a tento prístup nemusí byť umožnený. Inou možnosťou je softvérové meranie spotreby. Existuje mnoho aplikácií schopných merať spotrebu v reálnom čase. V rámci tejto práce bude meranie spotreby realizované pomocou knižnice PAPI.

Spotrebu je teoreticky možné vypočítať na základe vzorca 2.1.

$$P = C \times V^2 \times f \tag{2.1}$$

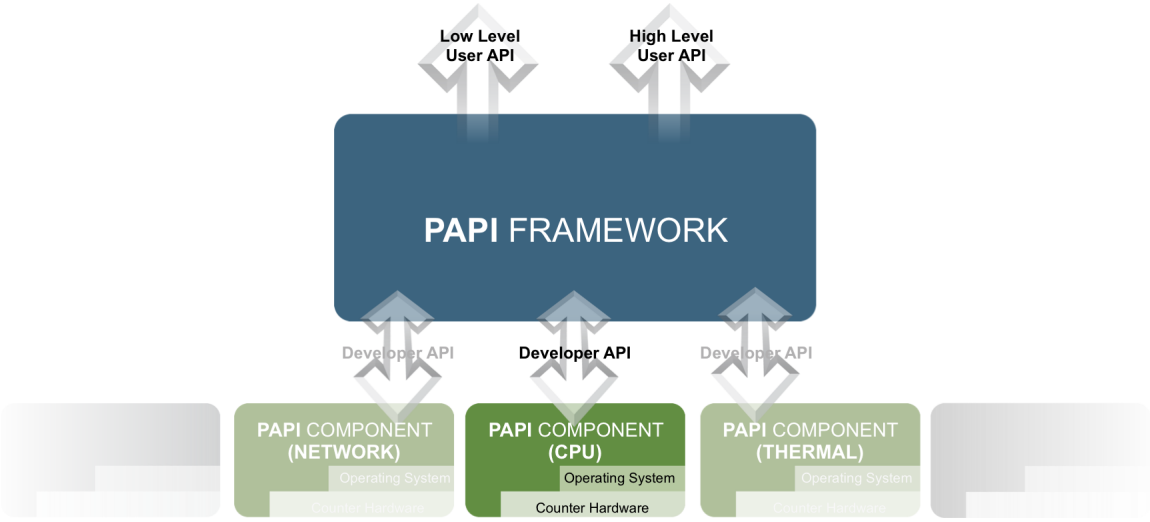
Kde P označuje vypočítaný výkon (V/A), C predstavuje kapacitu hradla (Coulomb/Volt), V označuje potenciál na hradle (Volt) a f je výška frekvencie (cykly/sekunda).

2.4.1 PAPI – Performance Application Programming Interface

Ide o knižnicu, ktorá špecifikuje štandardné rozhranie pre programovanie aplikácií (API) s možnosťou prístupu k hardvérovým počítadlám (counterom) výkonu, ktoré sú súčasťou väčšiny moderných procesorov. Počítadlá sú tvorené malým súborom registrov, do ktorých sa počítajú udalosti. Udalosť je chápaná ako výskyt špecifického signálu vzťahujúceho sa k funkcií CPU. Pomocou týchto údajov je možné ladiť aplikáciu, analyzovať jej výkonnosť, atď.

Architektúra PAPI pozostáva z dvoch rozhraní, pomocou ktorých je zabezpečený prístup k hardvérovým čítačom. Prvým je vysokoúrovňové rozhranie, ktorým je možné vykonávať

jednoduché merania. Toto rozhranie je plne programovateľné. Nízko úrovňové rozhranie poskytuje druhú možnosť prístupu k hardvérovým čítačom. Dovoľuje užívateľovi lepšie špecifikovať toto rozhranie vzhľadom k jeho potrebám. Pracuje s hardvérovými udalosťami, ktoré sú zoskupené do skupiny udalostí – `EventSet`. Skupina špecifikuje použitie počítadiel a rovnako je plne programovateľná.



Obr. 2.4: Architektúra PAPI (prevzaté z [12]).

Pomocou rozhrania nízkej úrovne je možná veľmi efektívna implementácia, ktorá poskytuje presnejšie a podrobnejšie výsledky meraní. Vysokoúrovňové rozhranie ponúka jednoduchú možnosť spustiť, zastaviť a čítať konkrétne udalosti. Rozhranie PAPI zobrazuje Obr. 2.4.

PAPI API je navrhnuté tak, aby bolo nezávislé na platforme. Využíva rovnaké rutiny s podobnými zoznamami argumentov pre prístup a riadenie počítadiel na každej architektúre. Zámerom rozhrania je možnosť použitia rovnakého zdrojového kódu s prípadnými drobnými úpravami, ktorý bude počítať podobné/porovnateľné skutočnosti po spustení na rôznych platformách. Problémom môže byť nepodporovanie udalostí inou platformou [12].

V rámci našej práci bude PAPI využívané primárne pre meranie spotreby jednotlivých komponentov, a tiež času potrebného pre výpočet daného problému. PAPI pre meranie spotreby využíva údaje z modulu `RAPL`.

RAPL obsahuje sadu čítačov, ktoré poskytujú informácie o spotrebovanej energií. Meranie je založení na softvérovom modeli. Spotreba je počítaná na základe informácií z hardvérových čítačov výkonu a na vstupno-výstupných modeloch. Na základe týchto informácií je možné kontrolovať spotrebu procesora a nastavenie chladenia podľa potreby (aktuálneho zaťaženia, zvoleného režimu) [15].

2.5 Výkonnosť

Určiť výkon počítača je možné rôznymi spôsobmi. Každý spôsob sa môže zameriavať na výkon z rôzneho hľadiska. Ide napríklad o testovanie konkrétneho komponentu počítača, výpočtového problému, atď. Pre tieto účely existuje veľké množstvo programov, ktorých výstupom je dosiahnuté skóre. Problémom týchto výsledkov je to, že sú špecifické pre konkrétny program. Tento spôsob určenia výkonu je vhodný pre použitie na bežných počíta-

čoch. V prípade, že by sme týmto spôsobom chceli zistiť výkon servera alebo superpočítača, s veľkou pravdepodobnosťou by sme narazili na problém s nekompatibilným operačným systémom alebo s neschopnosťou využiť potenciál testovaného zariadenia.

Pre účely zistenia výkonnosti bola zavedená jednotka `FLOP/s` (Floating point operations per second). Udáva počet operácií, v plávajúcej desatinnej čiarke, ktoré je CPU schopný spracovať za jednu sekundu. Hodnota vyjadruje maximálny teoretický výkon počítača. Určiť `FLOP/s` môžeme výpočtom, alebo použitím špecializovaného softvéru. Výpočet je definovaný vzorcom 2.2.

$$\frac{FLOP}{sekunda} = pocet\ CPU \times \frac{pocet\ jadier}{procesor} \times frekvencia \times \frac{pocet\ operacii}{takt} \tag{2.2}$$

Zo vzorca je zrejmé, že na maximálny výkon bude vplývať niekoľko parametrov. Väčšina z nich by mala byť uvedená v špecifikácií procesora a základnej dosky (systému). Problematickým parametrom môže byť počet operácií vykonaných počas jedného taktu. Nemusí byť zrejmé, že od čoho sa odvíja tento parameter. Hodnota tohoto parametru je daná použitou inštrukčnou sada pre spracovanie `SIMD` (Single Instruction, Multiple Data), viď Tab. 2.1.

Architektúra	**Inštrukčná sada**	**SP**	**DP**
Core 2/Nehalem	SSE2–128bit	8	4
Sandy/Ivy Bridge	AVX–256bit	16	8
Haswell/Broadwell	AVX2–256bit	32	16
Skylake Xeon	AVX512–512bit	64	32

Tabuľka 2.1: Počet operácií/takt v plávajúcej desatinnej čiarke (double/single precision) pre jednotlivé architektúry (prevzaté z [8]).

Tabuľka zobrazuje údaj pre architektúru, ktorá ešte nebola uvedená. V súčasnosti používajú AVX512 koprocesory Intel Xeon Phi, určené pre HPC (high performance computing).

2.6 Teplota

Súčasťou každého procesora je `DTS` (Digital Thermal Sensor). `DTS` meria teplotu každého jadra procesora a povrchu procesora. Na základe údajov, ktoré poskytuje je potom prispôsobovaný teplotný manažment, vďaka ktorému je zabezpečené, že procesor nepresiahne teplotný limit, na ktorý je navrhnutý. Hodnoty získané z týchto senzorov sú ukladané do špeciálnych registrov `MSR` (Model Specific Register). Každý z týchto registrov ukladá dáta z iného senzoru [2].

Pre meranie teploty je dostupné veľké množstvo nástrojov. Tieto nástroje získavajú údaje z vyššie spomínaných registrov, ktoré po spracovaní zobrazujú prostredníctvom užívateľského rozhrania. V tejto práci nebudú použité žiadne z dostupných nástrojov. Údaje o teplote budú získavané prostredníctvom súborov operačného systému pomocou vlastnej aplikácie. Vďaka tomu budeme schopný získavať dáta v požadovanej frekvencií a vo vhodnom formáte.

Kapitola 3

Paralelné architektúry

Architektúry paralelných počítačov je možné rozdeliť do dvoch základných skupín. Sú to architektúry s distribuovanou a so zdieľanou pamäťou.

Existujú dva prístupy k tvorbe paralelného programu. Filozofiou prvého prístupu je rozvetvenie výpočtu vo výkonovo kritických bodoch na viacero procesorov (zdieľaná pamäť). Tento prístup je možné implementovať pomocou knižnice `OpenMP` (viď Kap. 3.5).

Druhým spôsobom zabezpečenia paralelizácie je spustenie programu na viacerých procesorových uzloch. Zdieľanie dát medzi procesormi je zabezpečené zasielaním správ s dátami (distribuovaná pamäť). Pre implementáciu takýchto aplikácií je dostupná knižnica MPI (viď Kap. 3.4).

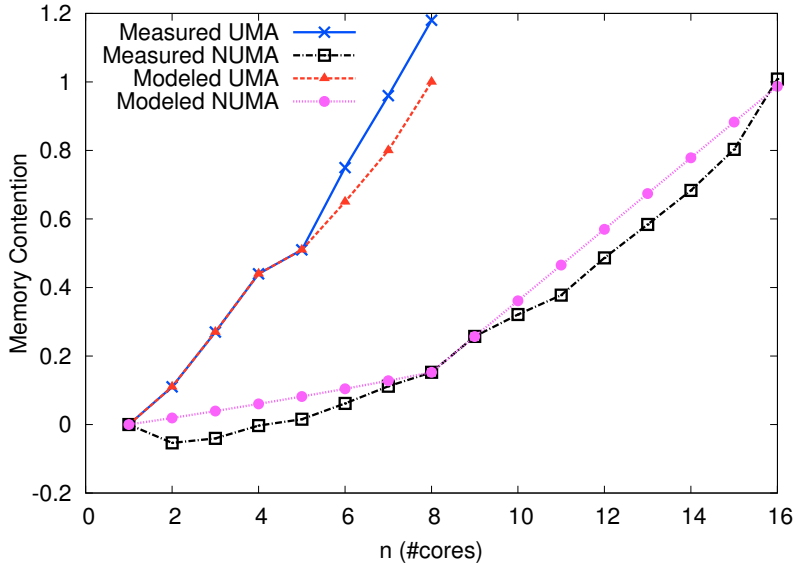
3.1 Zdieľaná pamäť

Prístup procesorov do pamäte je zabezpečený vysoko-rýchlostnou pamäťovou zbernicou. Zbernica je zdieľaná, vďaka čomu sú medzi sebou procesory schopné zdieľať údaje rýchlo a efektívne. Obmedzenie množstva spracovaných dát je dané šírkou pásma pamäťovej zbernice. Z toho dôvodu sa počet procesorov v tejto architektúre pohybuje v rozsahu 2–16, čo znamená že pamäťová škálovateľnosť je limitovaná. Pre minimalizovanie tohto obmedzenia je potrebná taká implementácia, ktorej snahou je čo najviac zredukovať synchronizáciu a častý prístup na rovnaké miesta v pamäti [18].

Tento typ architektúry je možné rozdeliť do dvoch skupín – s tzv. uniformným (rovnorodým), a z neuniformným (rôznorodým) prístupom do pamäte. Obe architektúry sa vzájomne odlišujú. Rozdiel je hlavne v ich škálovetaľnosti, teda v počte prepojených procesorov. Architektúra zdieľanej pamäte musí častejšie riešiť problém súčasného prístupu procesorov ku pamäti. V prípade, že sa dva rozdielne procesy (procesory) snažia pristúpiť k rovnakému bloku pamäte v jednom okamihu, vzniká situácia s názvom `memory contention`. Porovnanie týchto architektúr z hľadiska memory contention zobrazuje Obr. 3.1. Merania zobrazené v uvedenom obrázku boli vykonané pre maximálny počet jadier 8 (UMA) a 16 (NUMA). Zo získaných výsledkov je možné vyvodiť záver, že ak je pomer vlákna/jadrá väčší ako 4, v systémoch NUMA dochádza k výkonovým poklesom, ktoré sú spôsobené prepínaním kontextu [29].

3.1.1 UMA – Uniform Memory Access

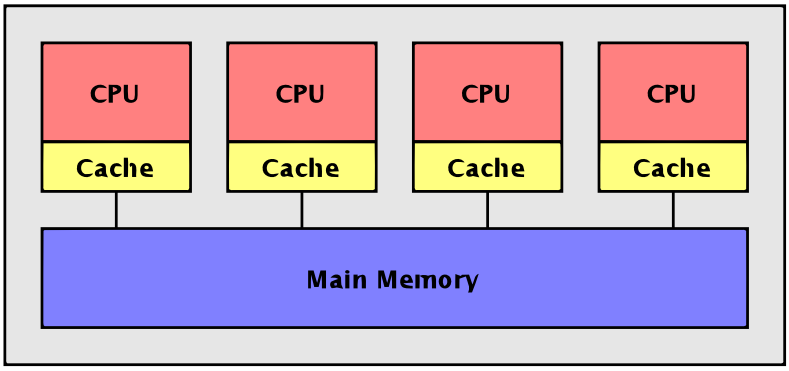
Z obrázku 3.2 je zrejmé, že procesory zdieľajú jednu spoločnú fyzickú pamäť. Každý z procesorov je ku pamäti pripojený pomocou zbernice. Všetky procesory sú rovnakého typu



Obr. 3.1: Porovnanie architektúr UMA a NUMA vo výskyte memory contention (prevzaté z [29]).

a prístupová doba ku každému pamäťovému regiónu je pre každý CPU rovnaká. Procesory využívajú vlastnú rýchlu vyrovnávaciu pamäť cache. Počítače tohoto typu sa niekedy označujú ako SMP (Symmetric Multiprocesor machine).

Tento model využívajú paralelné počítače. Sieť UMA je zložená z viacerých prístupových bodov a sieťových radičov, ktoré sú prepojené prostredníctvom širokopásmovej IP siete [26].

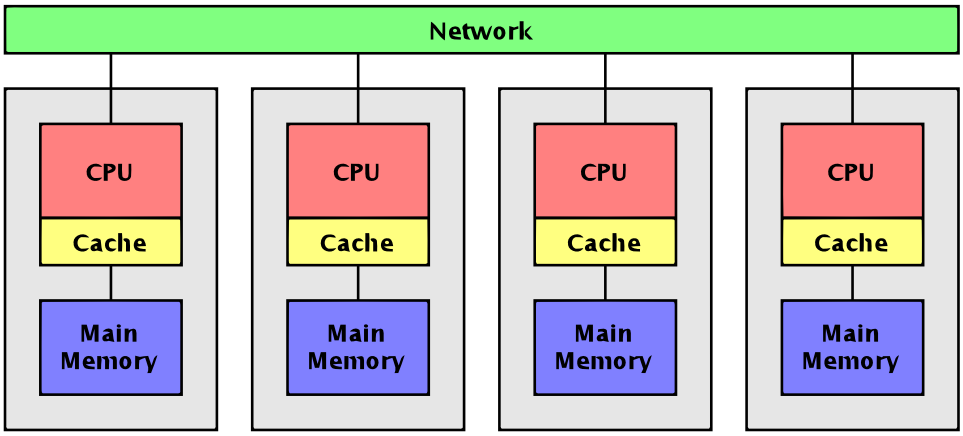


Obr. 3.2: Bloková schéma architektúry UMA (prevzaté z [21]).

3.1.2 NUMA – Non-Uniform Memory Access

Procesory rovnakého typu sú vzájomne prepojené do škálovateľnej siete, pričom každý z nich disponuje vlastnou pamäťou, viď Obr. 3.3. Hlavný rozdiel pri porovnaní s architektúrou distribuovanej pamäte je ten, že procesory architektúry NUMA nemajú vyhradenú pamäť iba pre seba. Vďaka tomu môžu pristupovať ku pamäti iného procesora. Vlastná zdieľaná pamäť sa potom označuje ako lokálna, cudzia zdieľaná pamäť ako vzdialená [23]. Prístup

do lokálnej pamäte je veľmi rýchly, avšak do vzdialenej, relatívne pomalý. Súbor lokálnych pamätí procesorov tvorí jeden logický adresný priestor.



Obr. 3.3: Bloková schéma architektúry (prevzaté z NUMA [21]).

3.2 Distribuovaná pamäť

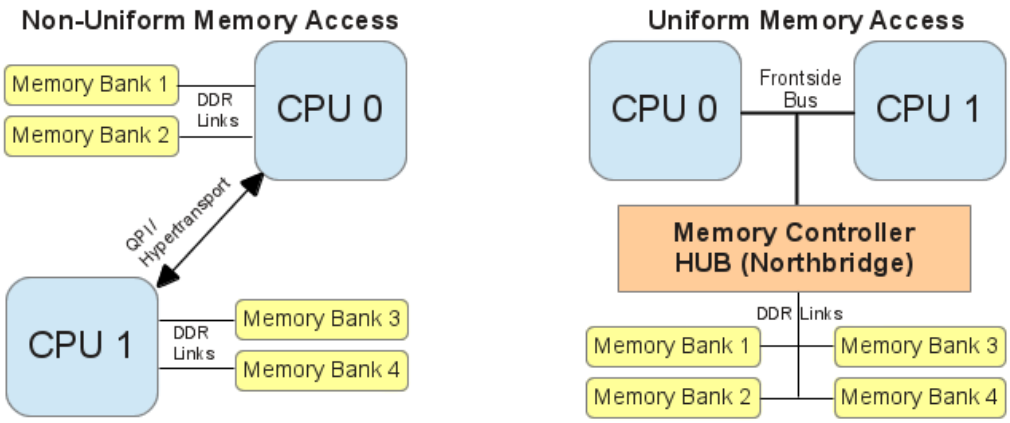
Počítače využívajúce tento typ architektúry je možné popísať ako niekoľko navzájom prepojených sériových počítačov (uzlov), ktoré spoločne pracujú na riešení výpočtového problému. Jednotlivé uzly majú veľmi rýchli prístup do vlastnej pamäte. Globálna, zdieľaná pamäť sa v týchto počítačoch nenachádza. Komunikácia medzi procesormi je preto zabezpečená tzv. komunikačnou sieťou. Na tejto vysoko-rýchlostnej sieti potom prebieha komunikácia medzi procesormi vo forme zasielania správ [18].

3.3 QPI – quick path interconnect

Ďalším krokom k zvýšeniu výkonu procesorov a ich priepustnosti, predovšetkým v oblasti viac-procesorových architektúr, bolo nasadenie technológie QPI. QPI je vysoko-rýchlostné rozhranie typu point-to-point, tvorené diferenciálnymi jednosmernými linkami. Využíva paketovú komunikáciu a „snoop“ protokol, ktorý je prispôsobený pre nízko-frekvenčné transakcie. Medzi podstatné vlastnosti QPI patrí dobrá spoľahlivosť, dostupnosť a prevádzkyschopnosť. Vďaka týmto vlastnostiam je táto technológia vhodná pre použitie v serverových systémoch, v ktorých zabezpečuje prenos dát medzi niekoľkými procesormi a vstupno-výstupným radičom [25].

Predchodcom QPI je FSB (Front-Side-Bus), teda technológia, ktorá slúži pre prepojenie CPU a čipovej sady (NB). Pri FSB je tok dát prenášaný cez jednu, zdieľanú, obojsmernú dátovú zbernicu. Neskôr bola zdokonalená pridaním dvoch nezávislých zberníc a vysoko-rýchlostných prepojení, vďaka čomu bola dosiahnutá vyššia priepustnosť. Teoretické maximum priepustnosti FSB je 6,4 GB/s (1,6 GT/s). QPI ovplyvnilo dnešnú architektúru procesorov takým spôsobom, že bola vyvinutá nová koncepcia s názvom Intel QuickPath Architecture. Táto architektúra sa odlišuje v tom, že radič pamäte je integrovaný do procesora. Spojenie radiča a pamäte je zabezpečené prostredníctvom QPI. Prínosom tejto technológie je vyššia priepustnosť, ktorá sa v súčasnosti pohybuje na úrovni 19,2 – 38,4 GB/s (4,8

– 9,6 GT/s), a menšie oneskorenie pri komunikácií, medzi jednotlivými komponentmi [1]. Rozdiel medzi architektúrami FSB a QPI zobrazuje obrázok 3.4.



Obr. 3.4: Vľavo architektúra NUMA využívajúca QPI, vpravo architektúra UMA s koncepciou FSB (prevzaté z [16]).

3.4 MPI

MPI (Message Passing Interface) je špecifikáciou pre vývojárov a používateľov knižníc na preposielanie správ. MPI vytvára štandard pre tvorbu programov, ktoré využívajú knižnice zasielania správ. Hlavnými bodmi pri vývoji tohto rozhrania boli – efektivita, prenositeľnosť, použiteľnosť a flexibilita. Rozhranie je špecifikované (jeho funkcie) pre jazyky C/C++ a Fortran.

Po vzniku bolo MPI cielené na architektúru s distribuovanou pamäťou. V súčasnosti je dostupná implementácia aj pre architektúry so zdieľanou pamäťou. Použitie MPI je možné takmer na každej architektúre.

Hlavnou výhodou MPI je dobrá prenositeľnosť a univerzálnosť. Program, využívajúci MPI, je platformovo nezávislí vďaka použitiu jednotného rozhrania [11]. Od začiatku vývoja vzniká paralelný program. Nevýhodou môže byť náročnejšie programovanie, ktoré vyžaduje lepšiu predstavivosť programátora.

3.4.1 Model zasielania správ

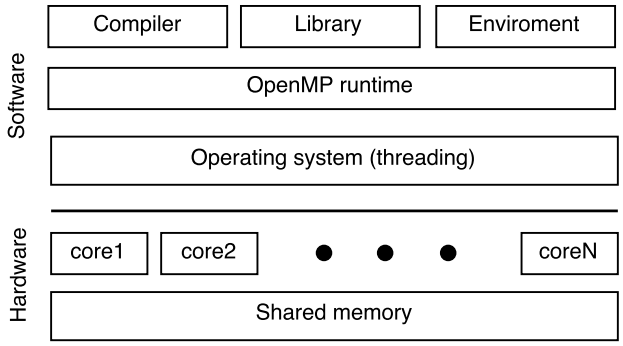
Paralelný výpočet sa skladá z niekoľkých procesov, pričom každý spracováva lokálne dáta. Procesy pristupujú iba do vlastnej pamäte a nemajú prístup do pamäte iného procesoru. Zdieľanie údajov je zabezpečené pomocou výmeny správ. Tá prebieha prijímaním a odosielaním údajov medzi procesmi.

Výhodou tohto modelu je jeho všeobecnosť. Z jeho využitím je možné implementovať skoro všetky paralelné algoritmy. Programy tohto typu umožňujú veľkú kontrolu nad tokom a umiestnením dát.

Nevýhodou je explicitné programovanie zasielacieho modelu.

3.5 OpenMP

OpenMP (Open Multi-Parallelism) je aplikačné rozhranie (API), ktoré poskytuje platformovo nezávislí programovací model, zameraný na programovanie aplikácií pre viacprocesorové systémy, so zdieľanou pamäťou. Tento model je definovaný pre programovacie jazyky C/C++ a Fortran. Na najnižšej úrovni ho tvorí sada direktív kompilátora, runtime knižníc a environmentálnych premenných, ktoré ovplyvňujú spôsob fungovania programu. Do vývoja tohto API sú zapojené významné softvérové a hardvérové spoločnosti a tiež množstvo výpočtových stredísk zameraných na vysokovýkonné paralelné výpočty a výskumnú činnosť [7].



Obr. 3.5: OpenMP model (prevzaté z [17]).

Výhodou tejto knižnice je jej jednoduchosť. Pomocou malých zásahov do sériového programu je možné program paralelizovať, s tým že nie je potrebné pôvodný kód upravovať. Požiadavkou pre použitie OpenMP je architektúra so zdieľanou pamäťou. OpenMP API ilustruje obrázok 3.5.

Kapitola 4

Benchmarky

Prvý krok k tomu, aby mohlo prebehnúť testovanie a analýza, je návrh sady benchmarkov. Vzhľadom k tomu, že sa v súčasnosti využívajú viacjadrové procesory, bolo potrebné vytvoriť takú sadu benchmarkov, ktorá bude zohľadňovať dnešné trendy. Preto by mala sada testov pozostávať z benchmarkov spúšťaných na jednom vlákne, a tiež z takých, ktoré budú bežať na väčšom počte vlákien.

4.1 Analýza algoritmov

Maximálny výkon počítača je dnes ovplyvnený predovšetkým rýchlosťou procesora. V súčasnosti je výkon procesora výrazne vyšší ako výkon pamäte. Dôsledkom toho je, že pamäť nedokáže poskytnúť procesoru taký objem dát, ktorých spracovanie by procesor dokázalo využiť na maximum [19]. Výkonová medzera medzi procesorom a pamäťou ovplyvňuje celkový výkon počítača v závislosti druhu algoritmu. Predmetom skúmania bude fakt, že do akej miery je táto medzera ovplyvniteľná zmenou (znížením) frekvencie CPU pri rôznych algoritmoch.

4.1.1 Memory bound algoritmy

Jedná sa o výpočtový problém, ktorý je závislí na veľkom objeme dát. Celková rýchlosť výpočtu je potom ovplyvňovaná šírkou pamäťovej (RAM) zbernice a prístupovou dobou pamäte. Procesor sa dostane do situácie, v ktorej nie je schopný využiť svoj plný výpočtový potenciál, pretože radič pamäti nedokáže naplniť jeho požiadavky na dáta [19]. Akcelerácia výpočtových problémov tohoto typu, zvýšením frekvencie procesora, je veľmi problematická. Použitie vyššej pracovnej frekvencie CPU nemusí znamenať kratšiu dobu výpočtu.

Algoritmy tohto typu môžeme nájsť napríklad v rôznych benchmarkoch pamätí. Jedným z takýchto benchmarkov sa budeme zaoberať v rámci tejto práce, viď Kap. 4.3.2.

4.1.2 CPU bound algoritmy

Výpočtové problémy tohoto typu nie sú veľmi závislé na priepustností pamäte. Rýchlosť výpočtu je ovplyvnená najmä rýchlosťou procesora. Úlohy, ktoré sú compute bound, obsahujú veľký počet aritmetických inštrukcií a len obmedzenú veľkosť dát, s ktorými pracujú. Z toho vyplýva, že počet operácií na jednotku dát, je dostatočne vysoký na to, aby procesor nebol obmedzovaný prísunom dát z pamäti. Príkladom takéhoto algoritmu je násobenie

malých matíc alebo generovanie náhodných čísiel. Akcelerácia rýchlosti výpočtu je možná zvýšením frekvencie CPU.

4.2 Jednovláknové algoritmy

Štruktúra programu je štandardne založená na sekvenčnom výpočte. Vytvorený algoritmus, ktorý rieši daný problém, je realizovaný ako postupnosť za sebou idúcich inštrukcií. Inštrukcie sú vykonávané jedným procesorom. Programy tohoto typu budeme označovať ako jednovláknové. Znamená to, že výpočet takéhoto programu bude obsluhovať jeden procesor a jedno jadro (vlákno).

Existuje veľký počet algoritmov, ktoré by sa dali použiť ako benchmarky. Pre testovanie boli zvolené nasledujúce algoritmy:

- Ackermannova funkcia
- Násobenie matíc
- Výpočet čísla π
- Quicksort

Predpokladom je, že každý z algoritmov bude mať iné charakteristické vlastnosti v priebehu výpočtu. Samotné testovanie a porovnanie získaných výsledkov je popísané v experimentálnej časti práce, viď Kap. 5.

Pri implementácií algoritmov bolo snahou použitie čo najjednoduchšej implementácie. Výpočet násobenia matíc a čísla π je vykonávaný iteratívne. Benchmarky Ackermannova funkcia a quicksort využívajú rekurzívnu implementáciu. V implementácií násobenia matíc sú použité tri dvojrozmerné polia, alokované vo voľnom úložisku. Výpočet čísla π je realizovaný pomocou metódy `Monte Carlo`. Použité algoritmy sú popísané pseudo-kódmi, viď Algoritmy 1 – 4.

Algoritmus 1: Ackermannova funkcia

Input: m, n
Output: Result of Ackermann function

`ACKERMANN(`m, n`):`

if $m = 0$ **then return** $n + 1$
else if $n = 0$ **then return** `ACKERMANN(`$m - 1, 1$`)`
else return `ACKERMANN(`$m - 1$, `ACKERMANN(`$m, n - 1$`))`

4.3 Viacvláknové algoritmy

Motiváciou pre zavedenie paralelného počítania je predovšetkým zrýchlenie a zefektívnenie výpočtu rozsiahlych úloh. Praktické využitie nachádza v mnohých sférach, ako sú napríklad simulácie v reálnom čase, inžinierske a vedecké výpočty, atď.

Programovacie modely pre tvorbu paralelných programov boli popísané v predchádzajúcej kapitole (viď Kap. 3).

Algoritmus 2: Násobenie matíc

```
Data: m1, m2, m3, size
for i ← 0 to size do
    for j ← 0 to size do
        for k ← 0 to size do
            m3[i][j] ← m3[i][j] + m1[i][k] × m2[k][j]
        end
    end
end
```

Algoritmus 3: Výpočet čísla π – metóda Monte Carlo

```
Data: x, y, z, pi, cnt, iter
for i ← 0 to iter do
    x ← rand()
    y ← rand()
    z ← x × x + y × y
    if z ≤ 1 then
        cnt ← cnt + 1
    end
end
pi ← cnt ÷ iter × 4 × 100
```

Algoritmus 4: Quicksort

```
Data: tmp, pivot, j
Input: first, last, pArr
Output: Sorted array

QUICKSORT(first, last, pArr):

if last > first then
    pivot ← pArr[last]
    j ← first − 1
    for i ← first to last + 1 do
        if pArr[i] ≤ pivot then
            j ← j + 1
            tmp ← pArr[j]
            pArr[j] ← pArr[i]
            pArr[i] ← tmp
        end
    end
    QUICKSORT(first, j − 1)
    QUICKSORT(j + 1, last)
end
```

4.3.1 Násobenie matíc a quicksort

Pri výbere paralelných algoritmov bola zohľadnená efektivita paralelnej implementácie. Dôležitým faktorom bolo aj to, že do akej miery sa zvolené algoritmy používajú v praxi. Na základe týchto kritérií boli zvolené algoritmy – násobenie matíc a quicksort.

Násobenie matíc je jedným z výpočtov, ktoré sa v reálnych programoch objavujú relatívne často. Uplatnenie nachádza v rôznych odvetviach s rôznym významom. Samotný algoritmus je pomerne jednoduchý. Keďže riešime problém paralelného výpočtu, bolo potrebné zvoliť vhodnú implementáciu. Pri testovaní bude potrebné kontrolovať počet vlákien, ktoré budú realizovať výpočet. Vzhľadom k druhu problému a tomuto požiadavku je použitá implementácia, ktorá využíva OpenMP.

Zoraďovanie je tiež veľmi častým problémom, ktoré je v praxi potrebné riešiť. Z toho dôvodu bol zvolený benchmark zameraný na riešenie takéhoto problému. Zoraďovací algoritmus quick sort (viď. Alg. 4) patrí k jedným z najznámejších. Jeho výhodou je predovšetkým rýchlosť a jednoduchosť. Voľba tohoto algoritmu bola ovplyvnená práve jeho rýchlosťou a popularitou. Podobne ako pri násobení matíc, bude testovaná implementácia využívajúca OpenMP.

4.3.2 Parallel memory bandwidth – PMBW

Nástroj PMBW je tvorený sadou rutín napísaných v asembleri, ktorých účelom je meranie priepustnosti pamätí. Podporované sú aj moderné paralelné a viacjadrové systémy. Priepustnosť pamäti je jednou z kľúčových vlastností, ktoré vplývajú na výkon počítačového systému. Meranie priepustnosti pamäti poskytuje reálnejší pohľad na celkový výkon zariadenia, na rozdiel od benchmarkov, určujúcich počet operácií v plávajúcej desatinnej čiarke, používaných v minulosti. Je to spôsobené tým, že dnešné CPU pracujú omnoho rýchlejšie ako pamäť, z čoho vypláva, že pamäť nedokáže zásobovať CPU takým množstvom dát, ktoré by dokázal spracovať. Preto je dôležité aby bola priepustnosť pamäte čo najvyššia, obzvlášť pri práci s veľkým objemom dát.

Tento benchmark je tvorený sadou základných funkcií napísaných v Assembleri, vďaka čomu sa predchádza potrebe optimalizovať kód prekladačom. Funkcie sú implementované v jednoduchých vnorených cykloch, ktoré sú súčasťou každého spracovania dát (práce s dátami). V závislosti od architektúry testovaného zariadenia sú použité rôzne inštrukcie (MMX, SSE, AVX).

Najväčším prínosom PMBW je možnosť testovania paralelného prístupu k pamäti, s tým že sa postupne zvyšuje počet vlákien. Výsledky testov poukazujú na problém, ktorý súvisí s paralelnými a viac jadrovými systémami. A síce že šírka pásma pamäte nie je prispôsobená počtu jadier dnešných procesorov.

Testovanie pomocou PMBW prebieha v niekoľkých krokoch. Po spustení testu program zistí veľkosť pamäti RAM, v ktorej sa následne alokuje pole. Veľkosť poľa je daná najvyššou možnou mocninou čísla dva, ktorá sa zmestí do RAM. Po tomto kroku sa spustí niektorá s rutín viď Obr. 4.1, napísaných v Assembleri. Táto rutina sa volá v iterácií so zvyšujúcou sa veľkosťou poľa, ktorého počiatočná veľkosť je `1024` bajtov a zväčšuje sa až do maximálnej veľkosti, ktorá bola alokovaná. Po dosiahnutí maximálnej veľkosti sa test opakuje pre vyšší počet vlákien. Pri spustení testu je možné nastaviť jeho parametre. Sú to napr. minimálna/maximálna veľkosť poľa, počet vlákien a tiež benchmarky, ktoré sa majú otestovať [14].

```
// ScanRead64PtrSimpleLoop
for (uint64_t* p = array; p < array + an; ++p)
   uint64_t x = *p;
```

```
// ScanWrite64PtrSimpleLoop
uint64_t x = 0xC0FFEE00C0FFEE00;
for (uint64_t* p = array; p < array + an; ++p)
   *p = x;
```

```
// ScanRead64PtrUnrollLoop
for (uint64_t* p = array; p < array + an; ++p) {
   uint64_t x0 = *(p+0);
   uint64_t x1 = *(p+1);
   // ... 13 times
   uint64_t x15 = *(p+15);
}
// ScanWrite64PtrUnrollLoop omitted
```

```
// ScanRead64IndexSimpleLoop
for (size_t i = 0; i < an; ++i)
   uint64_t x = array[i];
```

```
// ScanWrite64IndexSimpleLoop
uint64_t x = 0xC0FFEE00C0FFEE00;
for (size_t i = 0; i < an; ++i)
   array[i] = x;
```

```
// ScanRead64IndexUnrollLoop
for (size_t i = 0; i < an; i += 16) {
   uint64_t x0 = array[i+0];
   uint64_t x1 = array[i+1];
   // ... 13 times
   uint64_t x15 = array[i+15];
}
// ScanWrite64IndexUnrollLoop omitted
```

Obr. 4.1: PMBW rutiny (prevzaté z [14]).

4.3.3 LAMMPS – Molecular dynamics simulator

Simulátor LAMMPS sa zaoberá riešením problémov v oblasti molekulárnej dynamiky. Konkrétne ide o modelovanie rôznych štruktúr a systémov, pri ktorých sú zohľadnené okrajové podmienky a rôzne silové pôsobenia na chemické väzby.

LAMMPS je možné využívať na bežných počítačoch, keďže je schopný bežať na jednom CPU. Primárne je však určený na viacjadrové a viac-procesorové zariadenia. Knižnica vyžaduje dostupný prekladač jazyka C++ a podporu MPI.

Princíp simulátoru by sa dal zjednodušene charakterizovať nasledovne: simulátor integruje Newtonove pohybové rovnice pre skupinu atómov, molekúl alebo mikroskopických častíc. Častice interagujú pomocou krátko/dlho dobých síl s rôznymi počiatočnými a okrajovými podmienkami. Pri použití na paralelných systémoch LAMMPS využíva techniku priestorového rozkladania pre rozdelenie domén simulácie na menšie 3D subdomény, z ktorých každá je priradená jednému procesoru. Najvyššia efektivita výpočtu je dosahovaná pri využití paralelizmu [4].

Kapitola 5

Experimentálne výsledky

Bolo implementovaných niekoľko benchmarkov, na ktorých prebiehala analýza vplyvu frekvencie procesora na jednotlivé faktory, viď Kap. 4. Benchmarky sú rozdelené do dvoch základných skupín. Prvá skupina je zameraná na testovanie jednovláknových aplikácií. V druhej skupine je predmetom skúmania aj vplyv počtu vlákien, na ktorých je spustený výpočet. Obsahom kapitoly je tiež analýza vplyvu frekvencie, na teplotu CPU.

5.1 Testovaný hardware

Testovanie prebiehalo na školskom serveri `superphi2.fit.vutbr.cz`, ktorý bol v rámci bakalárskej práce sprístupnený. Server je typu Supermicro 7048GR-TR, v ktorom sú osadené nasledovné komponeny:

- MB Supermicro X10DRG-Q
- 2×CPU Intel Xeon E5-2620v3
- RAM DDR4-2133 64 GB (4 channels)
- SSD Crucial 250 GB
- Ethernet 1 Gb/s
- GPU NVidia GTX 980 4 GB GDDR5
- 3 Intel Xeon Phi
- PSU 2×2000 W

Maximálny teoretický výkon servera vypočítaný, pre frekvenciu 2,4 GHz, podľa vzorca 2.2, činí 460 GFLOP/s (presnosť `double`). V rámci tejto práce bol výkon overený pomocou benchmarku `Linpack`. Maximálna hodnota, ktorá bola dosiahnutá je rovná 390 GFLOP/s. Program Linpack dokázal reálne využiť približne 85 % výkonu z teoretického maxima.

5.2 Metodika testovania

Všetky benchmarky sú napísané v programovacom jazyku `C`. Binárne súbory boli vytvorené prekladom na školskom serveri, pomocou prekladača `GCC`, s výnimkou nástroja PMBW a LAMMPS. Pre tieto dva benchmarky sú binárne súbory dostupné na webových stránkach.

Pred začiatkom každého testu prebiehala kontrola, že či sa server nachádza v stave bez záťaže. Zmena frekvencie procesora bola vykonávaná použitím utility `cpupower`. Príklad zmeny frekvencie:

```
cpupower frequency-set -f 2400000
```

Hodnota frekvencie je uvádzaná v jednotkách `kHz`. Frekvencia procesora môže byť týmto spôsobom nastavovaná iba v určitom rozsahu (1,2 – 2,4 GHz) a po určitom kroku. Maximálna pracovná frekvencia testovaného procesora je podľa špecifikácie výrobcu až 3,2 GHz. Nastavenie tejto frekvencie popísaným spôsobom nie je možné. Procesor je schopný využívať túto frekvenciu iba v prípade použitia technológie Turbo, ktorá nie je dostupná pri nastavení konkrétnej frekvencie. Aby bolo možné dosiahnuť túto frekvenciu, tak je potrebné nastaviť `governor` na položku `ondemand`. Znamená to, že CPU prispôsobuje svoju frekvenciu aktuálnym potrebám a vyťaženiu. Vzhľadom k testovaniu benchmarkov je takéto chovanie problémové, pretože počas výpočtu môže dochádzať k zmenám frekvencie a výsledky spotreby jednotlivých komponentov nebudú v rovnakom vzťahu, vzhľadom k frekvencií CPU.

Na základe požiadaviek testov prebiehalo spúšťanie benchmarkov na určitom počte výpočtových vlákien. Bindovanie vlákien na jadrá procesora zabezpečoval príkaz `taskset`. Samotné spustenie potom prebiehalo nasledujúcim spôsobom:

```
taskset -c 0-5 ./benchmark
```

Získavanie údajov o spotrebe výpočtu zabezpečoval program, ktorý využíval knižnicu PAPI. Záznam informácií o spotrebe prebiehal každých 500 ms. Meranie čistého času výpočtu je implementované do každého benchmarku a tvorí jeho výstup. Výstupy logu spotreby a samotného benchmarku boli presmerované do súboru a následne vložené do tabuľkového kalkulátora, kde sa ďalej spracovali.

Získaná spotreba energie benchmarku je v tejto práci označovaná ako celková energia. Aby bolo možné porovnať spotreby benchmarkov, a ich charakter, bolo potrebné vypočítať tzv. „čistú“ spotrebu. Ide o množstvo energie, ktorá je potrebná pre výpočet daného problému, s tým, že sa vylúči energia, ktorú benchmark nespotreboval. Pre zistenie množstva energie, ktorú spotreboval daný výpočet, je potrebné odčítať energiu, ktorá by bola spotrebovaná v kľudovom stave (idle). Spotreba bez záťaže bola odmeraná pre každú frekvenciu. Dĺžka merania trvala päť minút. Z nameraných údajov sa určili priemerné hodnoty, ktoré zobrazuje tabuľka 5.1. Údaje udávajúce množstvo energie sú v jednotkách `Joule`. Platí to pre všetky grafy a tabuľky nachádzajúce sa v tejto kapitole (ak nie je uvedené inak).

Všetky benchmarky boli testované pri nastavení troch rôznych frekvencií procesora a to 1200, 1800 a 2400 MHz. Pri viacvláknových aplikáciách sa navyše menil aj počet výpočtových vlákien. Namerané údaje sú v práci zhrnuté do tabuliek a grafov.

f [MHz]	Package0 [W]	Package1 [W]	Dram0 [W]	Dram1 [W]
1200	16.0	15.9	14.0	9.3
1800	17.4	16.7	14.1	9.4
2400	18.4	17.2	14.1	9.4

Tabuľka 5.1: Priemerná spotreba superphi2 v idle.

5.3 Jednovláknové algoritmy

Benchmarky v tejto kapitole dokážu využiť iba jeden procesor a jedno jeho jadro. Na testovanom serveri sú k dispozícií dva procesory. Ak by testovanie benchmarkov prebiehalo

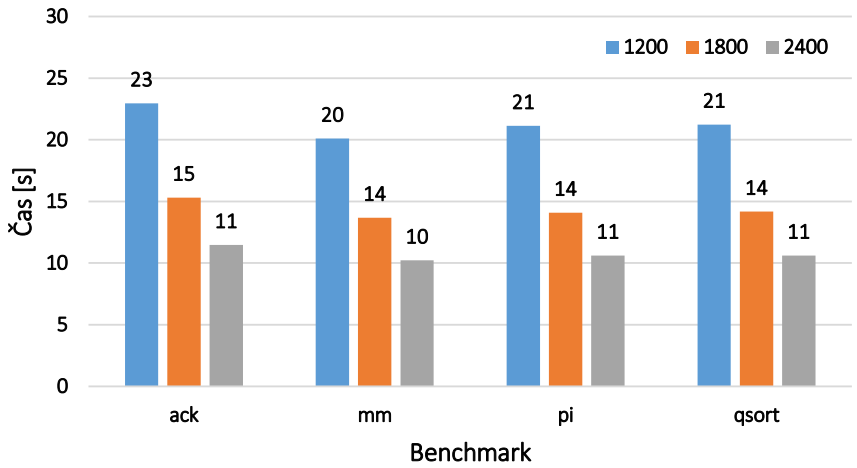
bežným spustením binárneho súboru, bolo by potrebné analyzovať údaje z oboch procesorov, pretože operačný systém môže prepínať úlohu medzi jednotlivými vláknami CPU. Toto správanie je možné potlačiť priradením úlohy na konkrétne vlákno procesora. Operačný systém Linux poskytuje prostriedky pre docielenie požadovaného chovania. Všetky jednovláknové benchmarky boli preto spustené príkazom

```
taskset -c 0 ./benchmark,
```

ktorým bolo zabezpečené, že bude úloha bežať vždy na vlákne 0. Znamená to, že bude využívané jadro 0 procesoru 0. Vďaka použitiu takejto konvencie spúšťania benchmarkov nie je potrebné zaoberať sa údajmi procesora, ktorý nie je používaný. Keďže je v architektúre použitého procesora operačná pamäť pripojená priamo ku CPU, tak sa v získavaní údajov môžeme obmedziť na jeden pamäťový modul.

Vstupné parametre týchto benchmarkov boli zvolené tak, aby bola ich doba výpočtu, pri nastavení na najvyššiu frekvenciu, približne rovnaká – 10 s. Pre výpočet Ackermannovej funkcie to znamená, že boli parametre `m` a `n` nastavené na hodnoty $m = 2$, $n = 25000$. Benchmark násobenie matíc, rieši problém násobenia dvoch matíc o veľkosti 1024^2, ktoré sú inicializované na náhodné hodnoty. Algoritmus výpočtu čísla π, určí toto číslo s presnosťou na 2 desatinné miesta pričom sa vykoná 350×10^6 iterácií. Benchmark quicksort zoraďuje pole o veľkosti 19×10^6 prvkov typu `integer`, ktoré sú náhodne vygenerované.

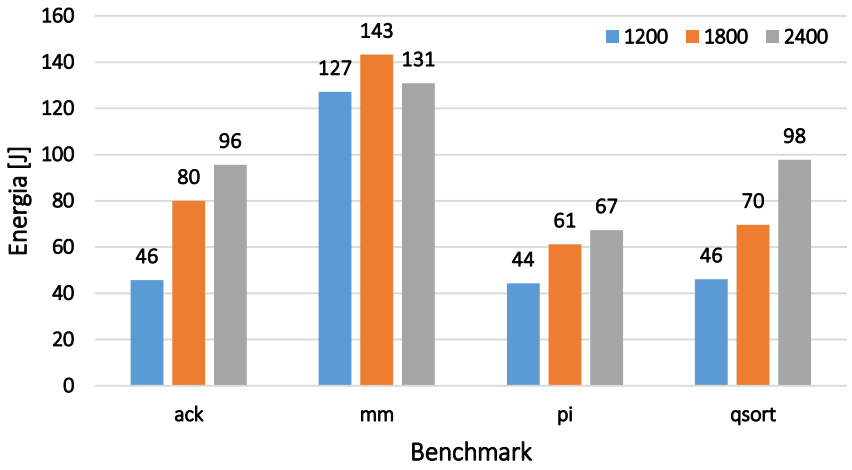
Rozdiely v čase výpočtu pri použití rôznych frekvencií zobrazuje graf 5.1. Akcelerácia doby výpočtu je priamo úmerná použitej frekvencií. Rozdiely medzi frekvenciami sú vo všetkých benchmarkoch približne v rovnakom pomere.



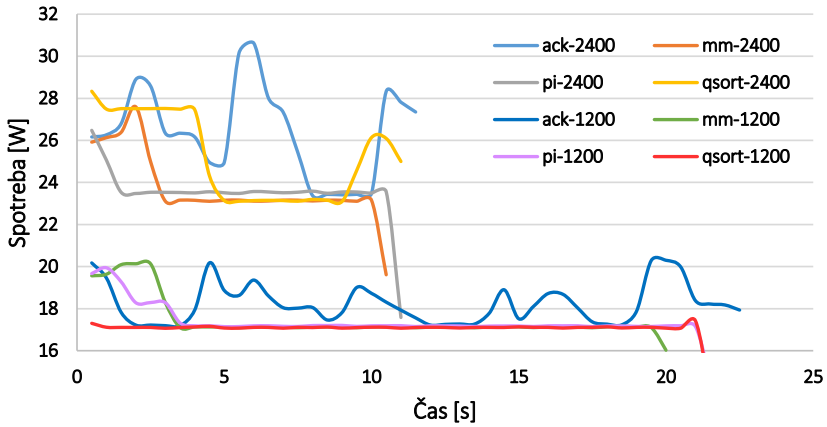
Obr. 5.1: Jednovláknové benchmarky – čas výpočtu benchmarkov pri nastavení CPU na rôzne frekvencie.

Z hľadiska „čistej“ spotreby energie, je podľa grafu 5.2, najvýhodnejšie použitie najnižšej frekvencie CPU. Z grafu sú viditeľné rozdiely v spotrebe medzi jednotlivými benchmarkmi. Zo spotreby je možné posúdiť, že do akej miery sa algoritmy odlišujú v ich zložitosti a tiež to, že rovnaký čas výpočtu nemusí znamenať rovnakú spotrebu.

Spotreba procesora by mala byť ovplyvňovaná predovšetkým jeho pracovnou frekvenciou, napätím a vyťažením. Porovnanie priebehu spotreby procesora pre jednotlivé algoritmy zobrazuje graf 5.3. Z grafu vyplýva, že výpočet Ackermannovej funkcie je pre CPU



Obr. 5.2: Jednovláknové benchmarky – „čistá“ spotreba energie CPU a RAM potrebná pre výpočet, pri nastavení CPU na rôzne frekvencie.



Obr. 5.3: Jednovláknové benchmarky – priebeh spotreby procesora pri nastavení frekvencie CPU na 1200/2400 MHz.

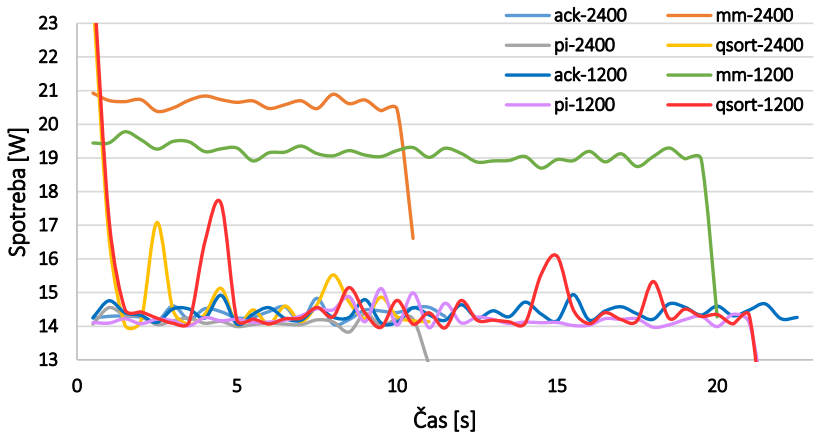
najnáročnejší. Rozdiely v spotrebe medzi jednotlivými benchmarkmi sú okrem Ackermanovej funkcie minimálne, ich priebehy sú lineárne.

Vplyv frekvencie procesora na spotrebu pamäti ilustruje graf 5.4. Podľa týchto údajov je zjavné, že najnáročnejší algoritmus na pamäť je násobenie matíc. Z grafu možno posúdiť, že rozdiel v spotrebe pamätí pri rôznych frekvenciách je takmer zanedbateľný.

Výsledky vykonaných testov dokazujú, že vyššia frekvencia procesora zrýchli výpočet, ale rovnako stúpne množstvo spotrebovanej energie. Spotreba „čistej“ energie ukazuje na to, že je vhodnejšie použitie najnižšej frekvencie.

5.4 Paralelné algoritmy

Kapitola sa zaoberá testovaním viacvláknových benchmarkov a teplotou CPU. Okrem vplyvu frekvencie na spotrebu a rýchlosť výpočtu je predmetom skúmania aj vplyv počtu vlákien na tieto faktory. Obsahom tejto časti je tiež testovanie priepustnosti pamäti a



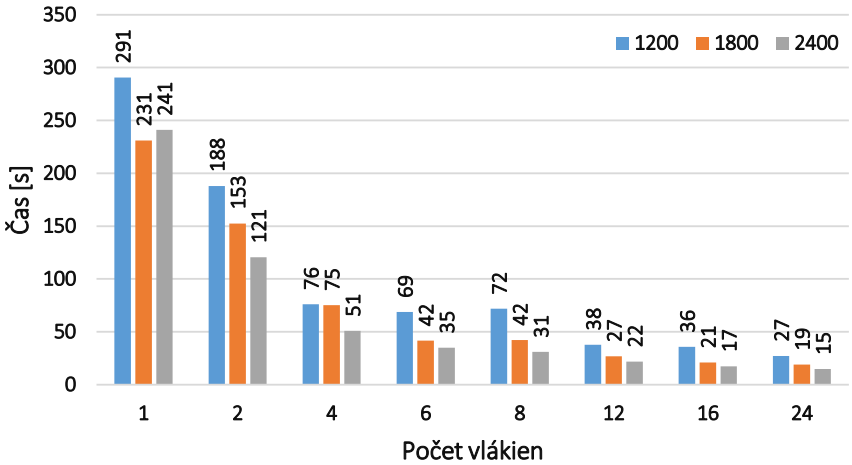
Obr. 5.4: Jednovláknové benchmarky – priebeh spotreby pamäte pri nastavení frekvencie CPU na 1200/2400 MHz.

jej závislosť od frekvencie CPU. Paralelizácia algoritmov je dosiahnutá použitím OpenMP 4.3.

5.4.1 Násobenie matíc

Tento benchmark rieši problém násobenia dvoch matíc (označené ako mm) o veľkosti 2048^2. V implementácií algoritmu sa nachádzajú tri dynamicky alokované dvojrozmerné polia (matice). Prvky dvoch sú inicializované na náhodne vygenerované desatinné čísla, ktoré sú uložené v textových súboroch. Prvky zvyšnej matice sú nastavené na nulovú hodnotu. Benchmark bol testovaný pri rôznom počte vlákien – 1, 2, 4, 6, 8, 12, 16, 24.

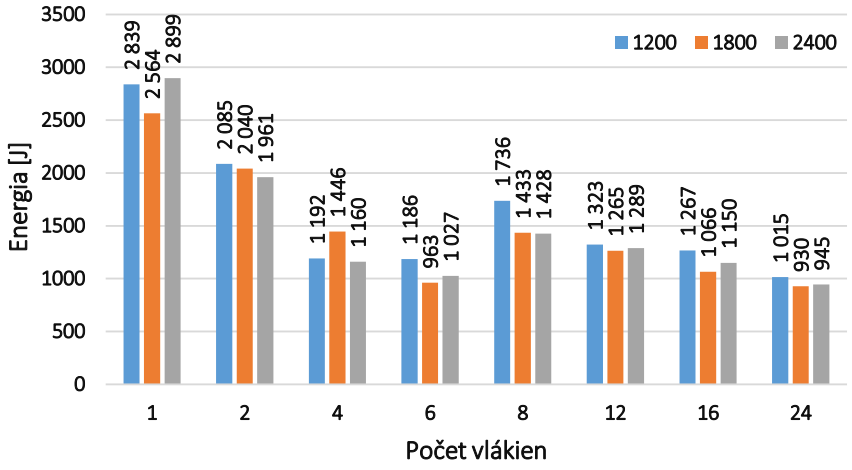
Z grafu 5.5 je viditeľné, že rýchlosť výpočtu bola ovplyvnená okrem frekvencie aj počtom vlákien. Akcelerácia spôsobená vyššou frekvenciou je v niektorých prípadoch až dvojnásobná.



Obr. 5.5: Paralelné násobenie matíc – celkový čas výpočtu závislý od frekvencie CPU a počtu vlákien.

Celková spotreba výpočtu zahŕňa energiu spotrebovanú procesormi (`package0`, `package1`)

a pamäťovými modulmi (`dram0`, `dram1`). Z grafu A.1b je zrejmé, že vzhľadom k celkovým prevádzkovým nákladom sa oplatí používať najvyššiu frekvenciu a všetky dostupné vlákna CPU. Pri pohľade na graf 5.6 možno usúdiť, že z hľadiska čistej spotreby nemusí byť v každom prípade najvýhodnejšie použitie najvyššieho počtu vlákien. Už pri počítaní na šiestich vláknach je spotrebovaná energia takmer na takej úrovni ako pri použití 24. vlákien. Odhliadnuc od času výpočtu, je potom jasné, že na tomto počte vlákien (6) je dosiahnutá najvyššia efektivita výpočtu.



Obr. 5.6: Paralelné násobenie matíc – „čistá“ energia, ktorú spotrebuje CPU a RAM počas výpočtu, vzhľadom k počtu vlákien a ku frekvencií.

Vývoj spotreby CPU a RAM pri rozdielnych frekvenciách, je viditeľný z grafov 5.7 a 5.8. Pri pohľade na spotrebu `package0` a `package1` je zjavné, že pre počet vlákien menší ako 8, je `package0` v stave idle. Rovnaké chovanie je možné pozorovať aj pri spotrebe pamäti. Príčinou takéhoto chovania je to, že pri spustení benchmarku dochádzalo k bindovaniu vlákien programu na jadrá CPU. Keďže procesor disponuje šiestimi jadrami, tak boli obsadzované najprv fyzické jadrá na jednom CPU. V prípade použitia viacerých vlákien ako je fyzických jadier jedného CPU, boli obsadené fyzické jadrá na druhom CPU, a až potom boli využívané hyper-threading vlákna. Zo spomínaných grafov možno usúdiť, že na spotrebu pamätí RAM nemá frekvencia CPU takmer žiadny vplyv. Priebehy spotreby procesorov a pamätí sa držia počas celej doby výpočtu na konštantnej úrovni. Krivky pre počet vlákien 1 a 2 nie sú kompletné, ich priebeh je naznačený šípkou.

Výsledky spotreby a času pre jednotlivé nastavenia popisuje tabuľka 5.2. Spotreba uvádzaná v tejto tabuľke je po odčítaní spotreby v režime idle. Zobrazuje teda „čistú“ energiu. Z výsledkov v tabuľke spotreby vyplýva, že z hľadiska spotreby procesorov a pamätí RAM, je najvýhodnejšie použitie strednej frekvencie (1800 MHz) a najvyššieho počtu výpočtových vlákien. Naopak najvyššia spotreba bola zaznamenaná pri najvyššej frekvencií, pri behu na jednom vlákne.

Záporné hodnoty v tabuľke pre položky `dram1` a `package1` sú spôsobené tým, že pri výpočte sa tieto komponenty nevyužívali, a ich priemerná spotreba bola menšia, ako odmeraná priemerná spotreba v režime idle.

Ďalšie údaje získané v súvislosti s týmto benchmarkom sú pripojené v prílohe A.2, B.1.

f [MHz]	Vlákien	Čas [s]	Package0 [J]	Package1 [J]	Dram0 [J]	Dram1 [J]	Celkom [J]	Celkom [W]
1200	1	290.6	369.70	51.98	2322.38	95.38	2839.44	9.77
1200	2	188.0	560.22	29.26	1484.27	11.59	2085.34	11.09
1200	4	76.2	485.82	35.83	621.48	49.34	1192.47	15.66
1200	6	68.8	662.81	-1.89	547.22	-22.06	1186.08	17.25
1200	8	71.8	446.32	237.55	572.84	479.20	1735.91	24.19
1200	12	37.8	357.21	404.65	295.28	265.37	1322.51	34.97
1200	16	35.8	328.72	352.64	335.81	250.21	1267.38	35.45
1200	24	27.2	281.87	312.73	225.96	194.43	1014.99	37.29
1800	1	231.0	559.33	51.75	1906.53	46.43	2564.04	11.10
1800	2	152.6	727.14	9.85	1226.20	77.21	2040.41	13.37
1800	4	75.3	756.48	22.71	618.93	48.10	1446.23	19.21
1800	6	41.7	635.93	-16.48	355.28	-11.58	963.14	23.08
1800	8	42.3	520.96	237.96	353.44	321.08	1433.44	33.86
1800	12	26.9	406.56	446.24	212.54	199.41	1264.74	47.06
1800	16	20.9	320.48	354.58	193.79	197.34	1066.18	50.91
1800	24	18.9	302.70	336.35	158.14	132.69	**929.88**	49.20
2400	1	241.0	1104.55	-141.07	1978.23	-43.20	**2898.50**	12.03
2400	2	120.6	994.59	-66.17	1003.78	29.02	1961.21	16.27
2400	4	51.0	788.17	-49.17	436.20	-15.30	1159.91	22.73
2400	6	35.0	766.27	-30.55	303.19	-11.66	1027.25	29.35
2400	8	30.9	669.38	280.91	276.96	200.27	1427.52	46.17
2400	12	21.8	471.60	480.57	194.97	141.48	1288.62	58.99
2400	16	17.5	393.46	416.67	176.78	163.57	1150.48	65.79
2400	24	14.9	346.01	365.21	123.23	110.58	945.03	63.57

Tabuľka 5.2: Paralelné násobenie matíc – spotrebovaná energia („čistá“) pri rôznych nastaveniach frekvencie CPU a počtu vlákien.

5.4.2 Quicksort

Implementácia tohoto benchmarku využíva knižnicu OpenMP. Benchmark rieši zoraďovanie poľa, v ktorom sú náhodne vygenerované hodnoty dátového typu `integer`. Tieto hodnoty sú uložené v súbore, ktorý je pri každom spustení načítaný, a prvky statického poľa o veľkosti 50×10^6, sú z neho inicializované. Testovanie benchmarku prebiehalo spustením na rôznom počte vlákien CPU – 1, 2, 4, 6, 8, 12, 16, 24.

Celková spotreba výpočtu zahŕňa energiu spotrebovanú procesormi (`package0`, `package1`) a pamäťovými modulmi (`dram0`, `dram1`). Z grafu A.1c je zrejmé, že vzhľadom k celkovým prevádzkovým nákladom sa oplatí používať najvyššiu frekvenciu a všetky dostupné vlákna CPU.

Pri pohľade na graf 5.10 možno usúdiť, že z hľadiska „čistej“ spotreby je najvýhodnejšie použitie jedného vlákna a najmenšej frekvencie. V prípade že je použitá najvyššia frekvencia, tak je najlepšou možnosťou použitie šiestich vlákien.

Priebeh spotreby procesorov a pamätí, pri spustení na rôznom počte vlákien, zobrazujú grafy 5.11 (1200 MHz), 5.12 (2400 MHz). Z priebehov je viditeľné, že pri spustení benchmarku dosahuje spotreba pamätí svoje maximum. So zvyšujúcim sa časom dochádza k znižovaniu spotreby pamätí. Spotreba procesorov počas výpočtu nie je lineárna. Krivky pre počet vlákien 1 a 2 nie sú kompletné a ich priebeh je naznačený šípkou.

Výsledky meraní pre tento benchmark sú uvedené v tabuľke 5.3. Grafy zobrazujúce priebeh spotreby procesorov a pamätí sú obsahom príloh A.3.

5.5 Optimalizácia

Ďalší spôsob, pomocou ktorého je možné dosiahnuť lepšiu efektivitu výpočtu, je použitie optimalizácie pri preklade. Prekladač GCC, použitý pre preklad benchmarkov poskytuje

f [MHz]	Vlákien	Čas [s]	Package0 [J]	Package1 [J]	Dram0 [J]	Dram1 [J]	Celkom [J]	Celkom [W]
1200	1	118.2	140.23	11.75	77.21	-11.95	**217.24**	1.84
1200	2	107.9	145.29	23.38	75.12	23.29	267.08	2.48
1200	4	58.0	180.74	11.03	78.45	14.17	284.38	4.90
1200	6	52.4	202.69	14.69	79.06	19.43	315.87	6.03
1200	8	42.8	88.04	107.84	124.16	103.73	423.77	9.90
1200	12	28.1	92.95	128.31	70.79	120.63	412.68	14.68
1200	16	17.3	102.70	124.05	70.64	97.03	394.42	22.74
1200	24	12.4	111.94	132.55	75.27	88.87	408.63	32.85
1800	1	78.9	217.16	16.54	62.15	19.88	315.73	4.00
1800	2	72.0	215.54	16.92	63.70	19.40	315.57	4.38
1800	4	39.7	232.66	14.13	71.61	16.78	335.18	8.44
1800	6	34.8	214.46	17.73	70.76	21.33	324.29	9.32
1800	8	22.0	120.81	137.99	103.32	80.44	442.57	20.14
1800	12	15.7	111.92	160.36	54.03	101.99	428.29	27.23
1800	16	10.8	126.43	143.46	77.71	65.69	413.29	38.38
1800	24	8.4	129.26	135.14	84.04	33.94	382.37	45.26
2400	1	61.5	310.05	3.47	47.70	10.08	371.31	6.03
2400	2	53.9	298.02	9.90	51.77	15.35	375.04	6.95
2400	4	29.1	289.52	10.87	57.27	13.32	370.98	12.76
2400	6	25.9	281.45	9.02	61.32	12.46	364.25	14.08
2400	8	18.4	160.26	166.92	102.83	63.89	**493.90**	26.79
2400	12	11.9	129.17	183.55	62.89	76.26	451.86	37.91
2400	16	9.1	130.47	152.35	49.84	69.03	401.69	43.96
2400	24	6.4	135.11	147.95	53.65	50.94	387.65	60.39

Tabuľka 5.3: Quicksort – spotrebovaná energia („čistá“) pri rôznych nastaveniach frekvencie CPU a počtu vlákien.

veľké množstvo optimalizácií. Vytvoriť optimalizovaný program je možné špecifikovaním konkrétnej optimalizácie, alebo využitím optimalizačnej sady. Tieto nastavenia sa určujú prepínačmi prekladača. Vplyv použitej optimalizácie na čas a spotrebu výpočtu bol testovaný pre benchmarky násobenie matíc a quicksort. Charakter oboch algoritmov je z hľadiska využitia procesora a pamätí odlišný. Tento fakt môže ovplyvňovať mieru optimalizácie.

Testované boli optimalizačné sady `O3`, `O2`, `O0`. Prepínač `-O0` vytvára binárny súbor bez optimalizácií, učený hlavne pre ladenie programu. Zvyšné dve sady sa zameriavajú na dosiahnutie najlepšieho výkonu programu. Prepínač `-O3` obsahuje viac optimalizácií pričom ale zahŕňa všetky optimalizácie sady `O2`. Efektivita optimalizácií v percentuálnom vyjadrení je zobrazená v grafoch. Jednotlivé stĺpce predstavujú zlepšenie výkonnosti v porovnaní s benchmarkom preloženým bez optimalizácií. Zlepšenie výkonnosti znamená kratšiu dobu výpočtu a nižšiu spotrebu. Frekvencie, pre ktoré bolo vytvorené porovnanie optimalizácií boli zvolené na základe výsledkov uvedených v Kap. 5.4.1 a 5.4.2, a to vzhľadom k najnižšej spotrebe energie. Skupiny stĺpcov zobrazujú výsledky pre daný počet vlákien. Testovanie prebiehalo na počte vlákien – 1, 2, 4, 6, 8, 12, 16, 24.

Rozdiely v optimalizáciách pre benchmark násobenia matíc popisuje graf 5.13. Údaje, ktoré obsahuje, boli odmerané pri frekvencií 1800 MHz. Z grafu vyplýva, že vhodnosť použitia konkrétnej optimalizačnej sady je závislá na počte výpočtových vlákien. Zlepšenie dosiahnuté optimalizáciami, má okrem behu na jednom vlákne, tendenciu klesať so zvyšujúcim sa počtom vlákien.

Porovnanie v grafe 5.14 zobrazuje rozdiely pri frekvencií 1200 MHz pre benchmark quicksort. Z výsledkov vyplýva, že pre tento benchmark, sú rozdiely v optimalizácií z hľadiska času výpočtu minimálne. Možno tiež usúdiť, že pri použití optimalizačnej sady `O3` je dosiahnutá menšia spotreba pri behu na ôsmych a šestnástich vláknach. V ostatných prípadoch je z hľadiska spotrebovanej energie výhodnejšie použitie optimalizačnej sady `O2`.

Na základe zistení v tejto kapitole možno usúdiť, že použitie optimalizácií môže mať

veľký vplyv na čas výpočtu a spotrebovanú energiu. Miera tohoto vplyvu je závislá od druhu použitého algoritmu a tiež od počtu výpočtových vlákien. Porovnanie optimalizácií pre všetky testované frekvencie sú zahrnuté v prílohách, viď Tab. B.3 a B.2.

5.6 PMBW

Zisťovanie vplyvu frekvencie CPU na rýchlosť pamäti prebiehalo pomocou nástroja PMBW popísaného v Kap. 4.3.2. Meranie priepustnosti pamäti bolo realizované pri behu na 1, 2, 4, 6, 12, 24 vláknach. Rýchlosť čítania a zápisu bola meraná pre výpočtový problém `IndexUnroll64` viď Obr. 4.1.

Z výsledkov uvedených v tejto kapitole vyplýva niekoľko záverov. Pri pohľade na niektorý z grafov možno usúdiť, že veľkosť poľa ovplyvňuje priepustnosť. Spôsobuje to fakt, že procesor pristupuje k dátam uloženým v rozdielnych vrstvách pamäte (L1, L2, L3, RAM), v závislosti od veľkosti problému. V grafickom znázornení je viditeľný schodovitý charakter kriviek. Ak sa pozrieme napríklad na graf 5.15b a budeme pozorovať krivku pre 6 vlákien, je viditeľné, že sa priepustnosť nemení až do momentu, kedy pole dosiahne veľkosť 2^{17} B. Ide o hraničnú veľkosť dát, ktoré sa zmestia do pamäte cache, úrovne L1. K ďalšiemu „prepadnutiu“ dochádza pri veľkosti poľa 2^{21} B (L2), k poslednému pri veľkosti 2^{24} B (L3). Od tohoto momentu procesor pristupuje k dátam, uloženým v hlavnej pamäti. Z uvedených grafov je zjavné, že rýchlosť tejto pamäte je niekoľkonásobne pomalšia ako pamäť cache.

Okamihy „prepadnutia“ do pomalších vrstiev pamäte (vyrovnávacej) však nesúvisia iba s ich veľkosťou. Sú závislé aj od počtu vlákien. Graf 5.16a ilustrujúci prístupovú dobu pamäte, jasne zobrazuje rozdiel vo veľkosti poľa pri dosahovaní týchto okamihov, pre jednotlivé počty vlákien. Príčina takéhoto chovania súvisí organizáciou pamäte CPU, viď Kap. 2.2.1. Pamäť L1 a L2 je vyhradená pre každé jadro, zatiaľ čo L3 je spoločná pre všetky jadrá CPU. Pri počte vlákien 12 dokáže procesor využiť všetku dostupnú pamäť cache. Na rozdiel od testu na jednom vlákne, kedy procesor môže využívať iba cache (L1, L2) jedného jadra. Z toho dôvodu dochádza k rozdielom vo veľkosti poľa, ktoré sa zmestí do jednotlivých úrovní cache, v závislosti od počtu vlákien. Pri porovnaní kriviek pre 12 a 24 vlákien, v tomto smere nedochádza k žiadnemu rozdielu. Je tomu tak z toho dôvodu, že cache úrovní L1 a L2 súvisí s počtom jadier (nie vlákien).

Výsledky tohoto benchmarku poukazujú na to, že počet vlákien je z hľadiska priepustnosti pamäti kľúčový. Najvyššie hodnoty sú dosahované v prípade využitia všetkých jadier a vlákien CPU 5.16 a 5.15. Spôsobuje to fakt, že pri viacvláknovom prístupe do pamäti sa využíva viac kanálov RAM čím narastá priepustnosť.

V grafoch zobrazujúcich priepustnosť pamäte pre rôzny počet vlákien, je možné pozorovať posun v začiatku kriviek, pre jednotlivý počet vlákien. Je to spôsobené tým, že program PMBW testuje priepustnosť pamäte pri rozdielnych veľkostiach polí pre každý počet vlákien.

Relatívny rozdiel v priepustnosti pamätí cache a hlavnej pamäte, vzhľadom k frekvencií, je minimálny. Z výsledkov je viditeľné, že použitím viacerých vlákien je možné dosiahnuť vyššiu priepustnosť. Ostatné grafy vytvorené z výsledkov benchmarku PMBW sú pripojené v prílohe, viď A.4, A.5, A.6.

Na základe zistení v tejto kapitole možno usúdiť, že priepustnosť hlavnej pamäti aj pamäti cache, sú závislé na frekvencií procesora. V článku [27] je popísané dosiahnutie podobných výsledkov, pri testovaní dvoj-procesorovej konfigurácie, procesoru Sandy Bridge-EP.

5.7 LAMMPS

Benchmark LAMMPS bol zvolený ako zástupca softvéru používaného v praxi. Jeho bližší popis je uvedený v Kap. 4.3.3. Dáta nachádzajúce sa v tejto kapitole, boli získané z dvoch rôznych simulácií. Rozdiel medzi nimi je predovšetkým v ich pamäťovej náročnosti. Výpočty úloh boli spustené paralelne, na všetkých dostupných vláknach CPU (24). Paralelizácia programu bola dosiahnutá pomocou MPI, jeho spustenie prebiehalo príkazom:

```
mpirun -np 24 ./benchmark
```

Hodnoty spotreby dosiahnuté pri výpočte týchto úloh zobrazujú tabuľky 5.4 (SPC/E) a 5.5 (FENE). Zo získaných výsledkov vyplýva, že vzhľadom k spotrebe je najvýhodnejšia frekvencia procesora 1800 MHz.

f [MHz]	Čas [s]	Package0 [J]	Package1 [J]	Dram0 [J]	Dram1 [J]	Celkom [J]	Celkom [W]
1200	77	913.01	1025.87	805.77	740.03	**3484.68**	45.26
1800	51	951.69	1024.79	595.18	534.35	**3106.00**	60.90
2400	38.5	1100.16	1129.03	486.41	463.60	3179.19	82.58

Tabuľka 5.4: „čistá“ energia spotrebovaná výpočtom benchmarku SPC/E (LAMMPS) – pamäťová náročnosť 86 MB.

f [MHz]	Čas [s]	Package0 [J]	Package1 [J]	Dram0 [J]	Dram1 [J]	Celkom [J]	Celkom [W]
1200	16.5	181.42	200.14	110.88	100.04	**592.48**	35.91
1800	11.5	186.14	203.81	78.63	74.45	**543.03**	47.22
2400	9.5	225.53	228.47	61.93	52.46	568.38	59.83

Tabuľka 5.5: „čistá“ energia spotrebovaná výpočtom benchmarku FENE bead/spring (LAMMPS) – pamäťová náročnosť 8,4 MB.

Z priebehu spotreby procesorov 5.18 je viditeľná závislosť spotreby od frekvencie. Spotreba oboch procesorov pri rovnakej frekvencií sa pohybuje na veľmi podobnej úrovni, zatiaľ čo rozdiely v spotrebe medzi jednotlivými frekvenciami sú omnoho výraznejšie.

V spotrebe pamätí pri rôznych frekvenciách je viditeľný iba malý rozdiel, v porovnaní s rozdielom medzi spotrebami procesorov. Výraznejší posun je možné pozorovať medzi spotrebou jednotlivých pamäťových modulov. Príčinou tohoto rozdielu môže byť využitie pamäťového modulu operačným systémom.

5.8 Teplota

Obsahom tejto kapitoly je testovanie vplyvu frekvencie CPU na jeho teplotu. Získavanie údajov o teplote bolo vykonávané pomocou vytvoreného programu. Algoritmus programu meral teplotu každých 500 ms a vypisoval ju na štandardný výstup. Údaje o teplote boli získavané čítaním súborov `temp*_input` s adresára `/sys/class/hwmon/hwmon*`. Identifikácia konkrétneho vstupu bola určená súborom `temp*_label`, nachádzajúcim sa v rovnakom adresári.

Meranie teploty pri rôznych frekvenciách prebiehalo pri vyťažení CPU benchmarkom Linpack. Priebeh teplôt procesorov (`package0`/`package1`) ilustruje graf 5.21. Percentuálny rozdiel v teplotách, pre jednotlivé procesory a ich jadrá, v porovnaní s najnižšou frekvenciou (1200 MHz) popisuje graf 5.20. Z grafu je viditeľné, že napriek použitiu dvojnásobnej

frekvencie CPU, je maximálny nárast teploty povrchu procesora necelých 15 %. Rozdiel v maxime teploty jadier dosahuje necelých 30 %. Na základe získaných výsledkov je možné usúdiť, že vplyv frekvencie na teplotu nie je veľmi dramatický.

5.9 Zhrnutie experimentov

V rámci tejto kapitoly bolo vykonaných niekoľko testov, zameraných na skúmanie vplyvu frekvencie procesora na čas výpočtu, spotrebu a teplotu. Testovanie bolo rozdelené na jednovláknové a viacvláknové problémy. Z výsledkov všetkých testovaných benchmarkov vyplýva, že použitie vyššej frekvencie v každom prípade skracuje dobu výpočtu.

Teoreticky je možné usúdiť, že z hľadiska celkových nákladov na spotrebu je najvýhodnejšie použitie najvyššej frekvencie CPU. Spôsobuje to fakt, že spotreba počítača je závislá hlavne od doby, kedy je zapnutý. Tento záver platí len v prípade, ak by bol počítač používaný iba pre vykonanie konkrétneho výpočtu, po ktorého dokončení, by bol okamžite vypnutý. V praxi je toto riešenie nepredstaviteľné. Z toho dôvodu je vhodnejšie riešiť spotrebu, ktorá nezahŕňa energiu potrebnú pre chod počítača. Použitím tohto prístupu je možné objektívnejšie posúdiť výhodnosť konkrétnej frekvencie a celkovej spotreby algoritmu.

Namerané údaje boli zhrnuté do tabuliek, ktoré zobrazujú percentuálne zlepšenie, času výpočtu a spotreby energie, v porovnaní s najnižšou frekvenciou (1200 MHz). Tabuľka 5.6 zobrazuje výsledky testov pri spustení algoritmov na jednom vlákne. Na základe údajov v tabuľke možno usúdiť, že z hľadiska zníženia spotreby je použitie vyššej frekvencie pre jednovláknové benchmarky nevhodné (s výnimkou benchmarku násobenie matíc). Naopak výsledky viacvláknových algoritmov, viď Tab. 5.7, ukazujú, že pri použití vyšších frekvencií je dosiahnutá podobná alebo ešte nižšia spotreba, ako v prípade použitia najmenšej frekvencie.

Zistenie, ktoré plynie z vykonanej práce ukazuje, že vhodnosť použitej frekvencie procesoru je z veľkej časti závislá na druhu výpočtového problému. Voľba vhodnej frekvencie je tiež závislá od preferencie na určitý faktor. Výpočet je realizovaný s ohľadom na najlepšiu spotrebu, alebo na najkratší čas výpočtu. Z preukázaných výsledkov vyplýva, že v určitých prípadoch dochádza k prínosu vyššej frekvencie, v oboch uvedených smeroch. Pre niektoré výpočtové problémy to ale neplatí.

Množstvo spotrebovanej energie je ovplyvnené aj spôsobom prekladu zdrojového textu programu. Testovaný bol prekladač `GCC` a optimalizačné sady `O0`, `O2` a `O3`. Pri použití optimalizácií sa dosiahnutá akcelerácia doby výpočtu a zníženie spotreby odvíjalo od použitého algoritmu a od počtu výpočtových vlákien.

Náplňou práce bola tiež analýza teploty procesora v závislosti na frekvencií. Zo získaných údajov vyplýva, že vplyv frekvencie na teplotu nie je veľmi výrazný, keďže sa server nachádza v klimatizovanej serverovni a o jeho sa stará šesť vysoko-otáčkových 12 cm ventilátorov. Najväčší zaznamenaný rozdiel v teplote povrchu procesora, pri porovnaní s najnižšou frekvenciou, dosiahol hodnotu 15 %. V porovnaní s akceleráciou času výpočtu a prípadným znížením spotreby je rozdiel v teplote zanedbateľný.

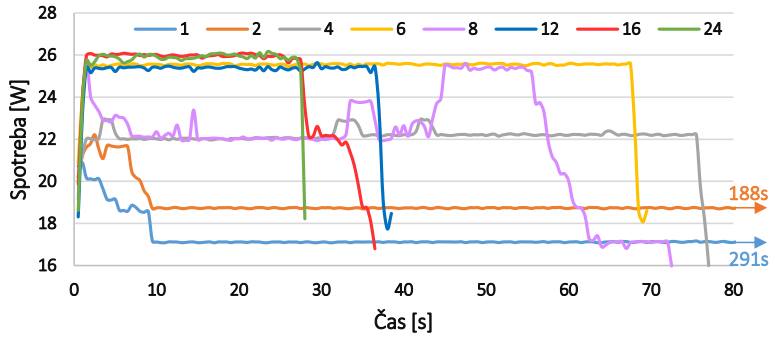
Z testov priepustnosti pamäti vyplýva, že frekvencia procesora ovplyvňuje predovšetkým rýchlosť cache procesora.

Benchmark	Čas 1,8 GHz	Spotreba 1,8 GHz	Čas 2,4 GHz	Spotreba 2,4 GHz
ack	33 %	-75 %	50 %	-109 %
mm	32 %	-13 %	49 %	-3 %
pi	33 %	-38 %	50 %	-52 %
qsort	33 %	-51 %	50 %	-112 %
priemer:	33 %	-44 %	50 %	-69 %

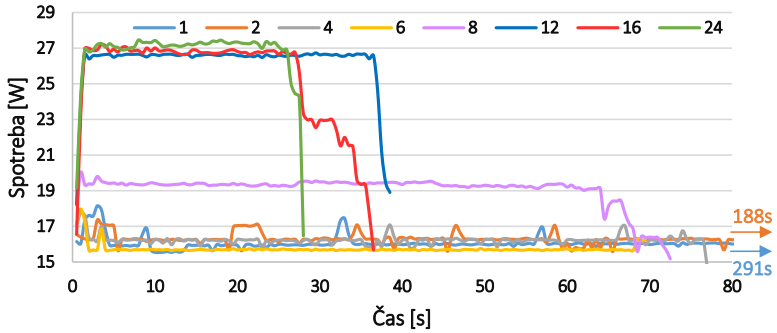
Tabuľka 5.6: Zhrnutie akcelerácie doby výpočtu a zníženia spotreby pre 1 vlákno.

benchmark		mm 1024		mm 2048		quicksort	
f [MHz]	Vlákien	Čas [s]	Spotreba [J]	Čas [s]	Spotreba [J]	Čas [s]	Spotreba [J]
1800	1	32 %	20 %	20 %	10 %	33 %	-45 %
1800	2	32 %	-1 %	19 %	2 %	33 %	-18 %
1800	4	32 %	17 %	1 %	-21 %	32 %	-18 %
1800	6	33 %	25 %	39 %	19 %	34 %	-3 %
1800	8	33 %	0 %	41 %	17 %	49 %	-4 %
1800	12	32 %	30 %	29 %	4 %	44 %	-4 %
1800	16	24 %	10 %	41 %	16 %	38 %	-5 %
1800	24	32 %	0 %	31 %	8 %	32 %	6 %
priemer:	–	31 %	13 %	28 %	7 %	37 %	-11 %
2400	1	49 %	37 %	17 %	-2 %	48 %	-71 %
2400	2	49 %	7 %	36 %	6 %	50 %	-40 %
2400	4	49 %	23 %	33 %	3 %	50 %	-30 %
2400	6	49 %	9 %	49 %	13 %	51 %	-15 %
2400	8	49 %	6 %	57 %	18 %	57 %	-17 %
2400	12	48 %	5 %	42 %	3 %	58 %	-9 %
2400	16	44 %	6 %	51 %	9 %	47 %	-2 %
2400	24	49 %	5 %	45 %	7 %	48 %	5 %
priemer:	–	49 %	12 %	41 %	7 %	51 %	-22 %

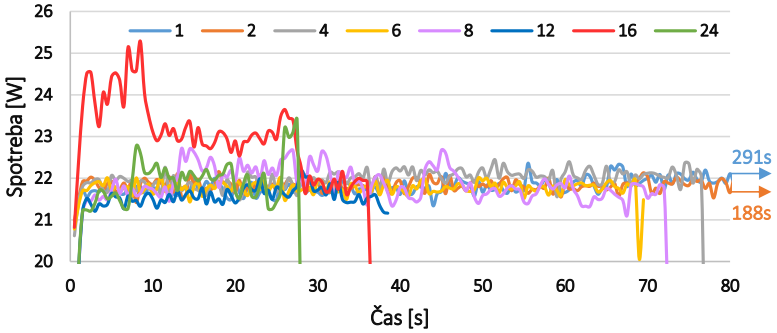
Tabuľka 5.7: Zhrnutie akcelerácie doby výpočtu a zníženia spotreby pre viacvláknové algoritmy.



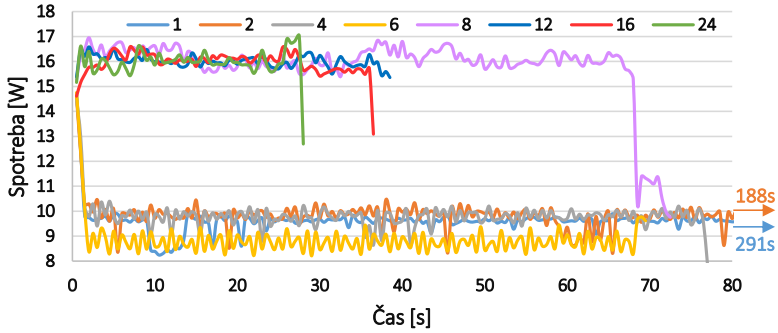
(a) Package0



(b) Package1

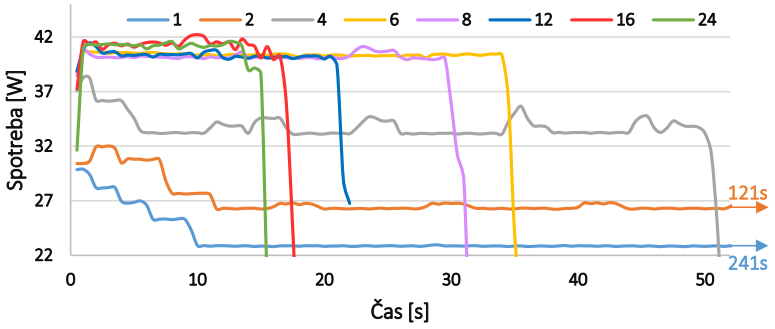


(c) Dram0

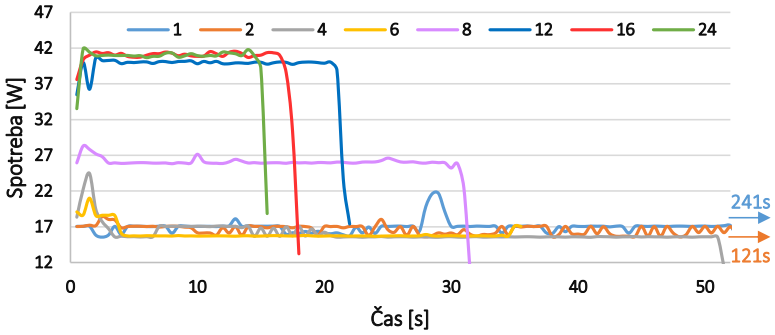


(d) Dram1

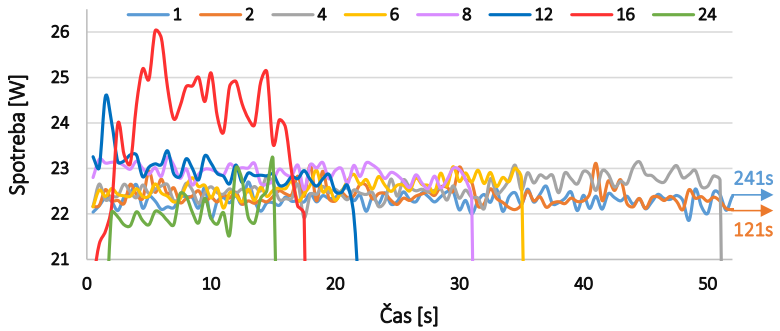
Obr. 5.7: Paralelné násobenie matíc – priebeh spotreby CPU a RAM pre rôzny počet vlákien, nastavenie frekvencie CPU na 1200 MHz.



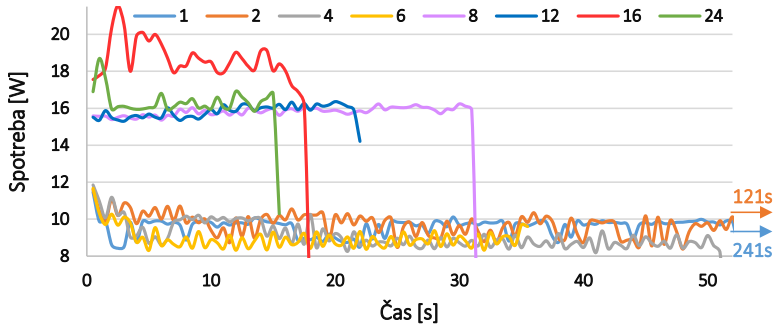
(a) Package0



(b) Package1

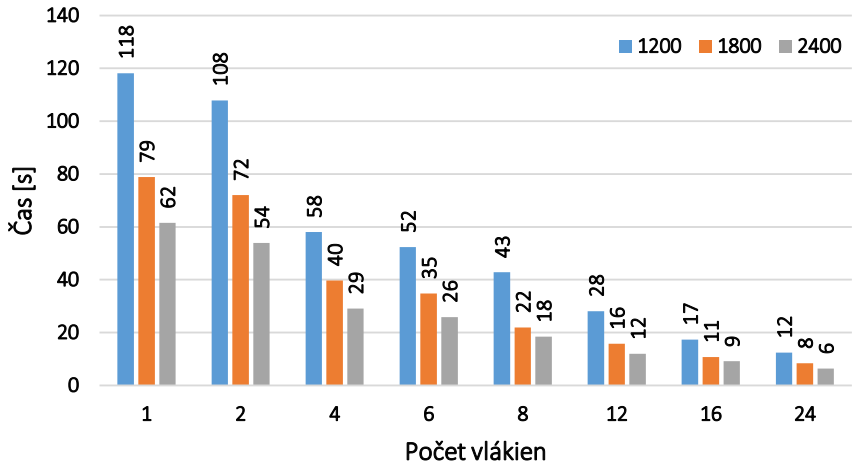


(c) Dram0

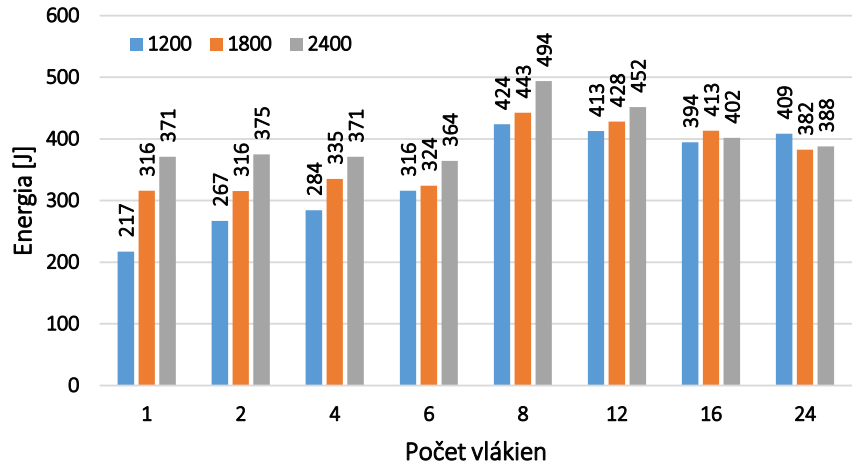


(d) Dram1

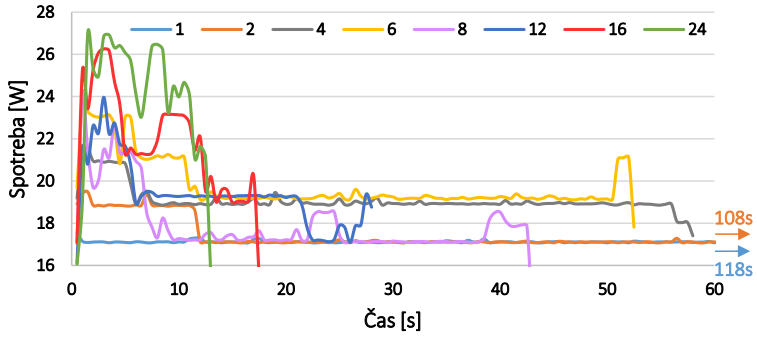
Obr. 5.8: Paralelné násobenie matíc – priebeh spotreby CPU a RAM pre rôzny počet vlákien, nastavenie frekvencie CPU na 2400 MHz.



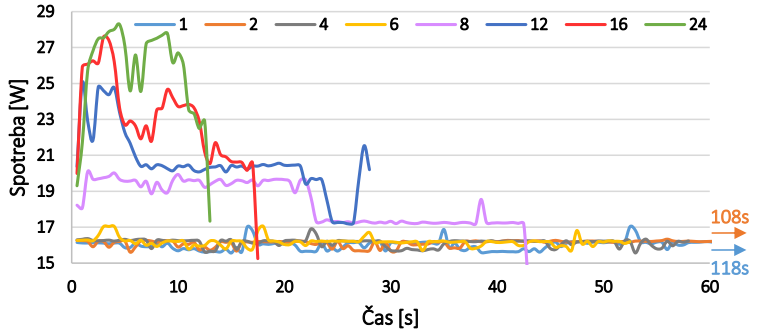
Obr. 5.9: Quicksort – celkový čas výpočtu závislý od frekvencie CPU a počtu vlákien.



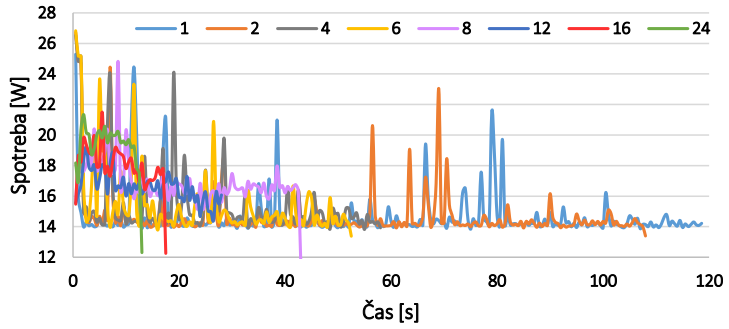
Obr. 5.10: Quicksort – „čistá“ energia, ktorú spotrebuje CPU a RAM počas výpočtu vzhľadom ku počtu vlákien a k frekvencií.



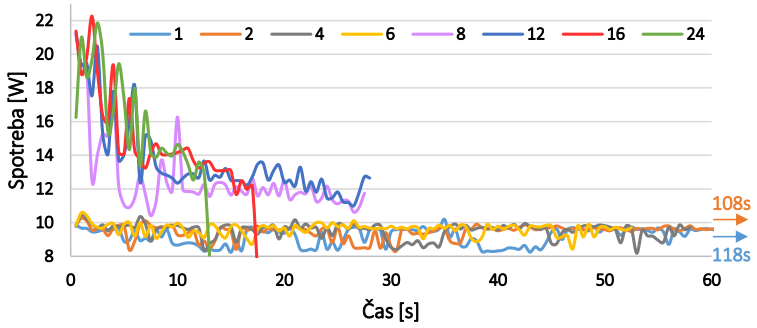
(a) Package0



(b) Package1

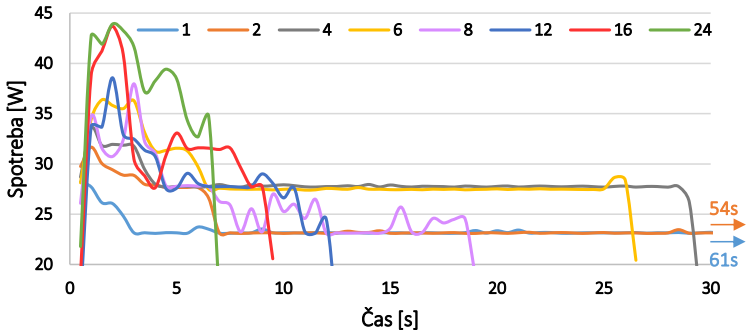


(c) Dram0

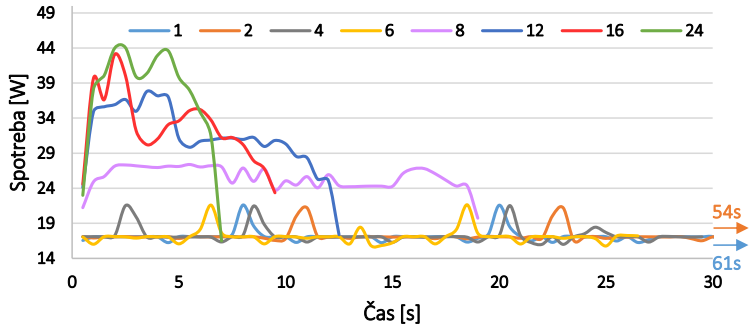


(d) Dram1

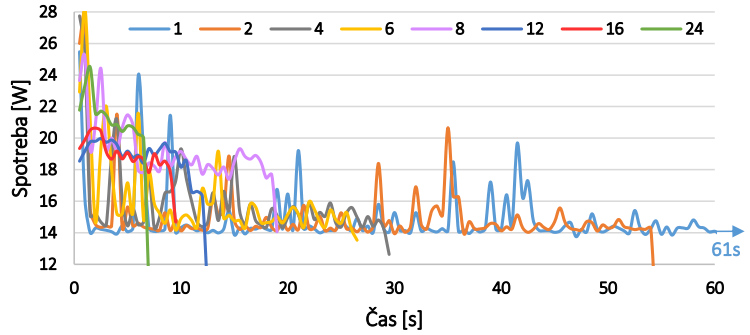
Obr. 5.11: Quicksort – priebeh spotreby CPU a RAM pre rôzny počet vlákien, nastavenie frekvencie CPU na 1200 MHz.



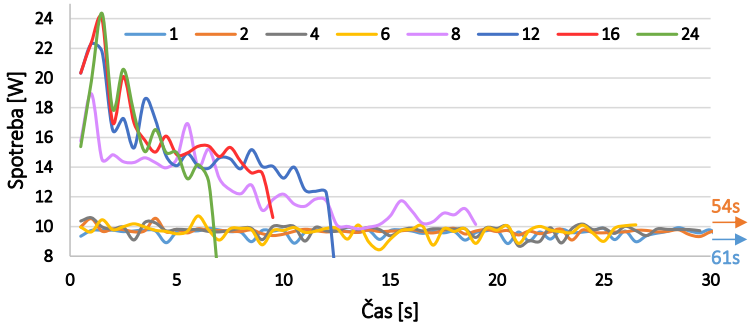
(a) Package0



(b) Package1

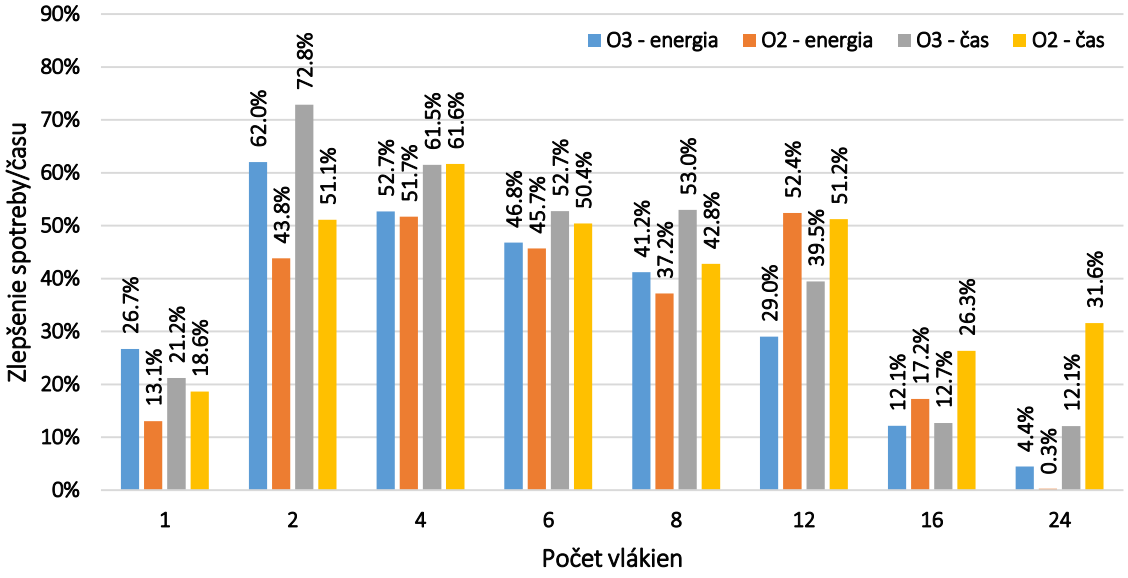


(c) Dram0

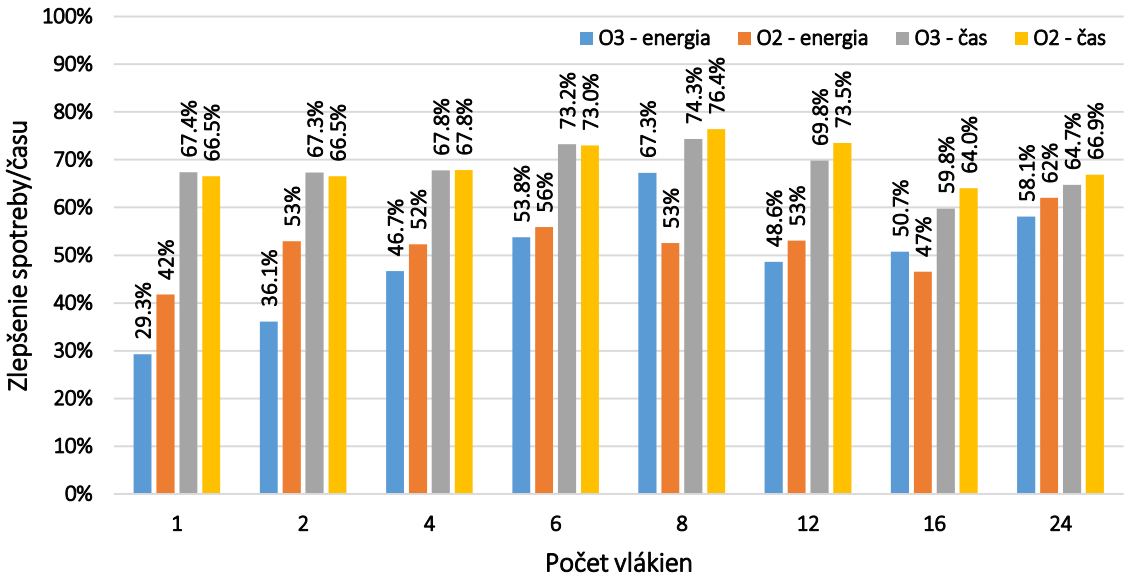


(d) Dram1

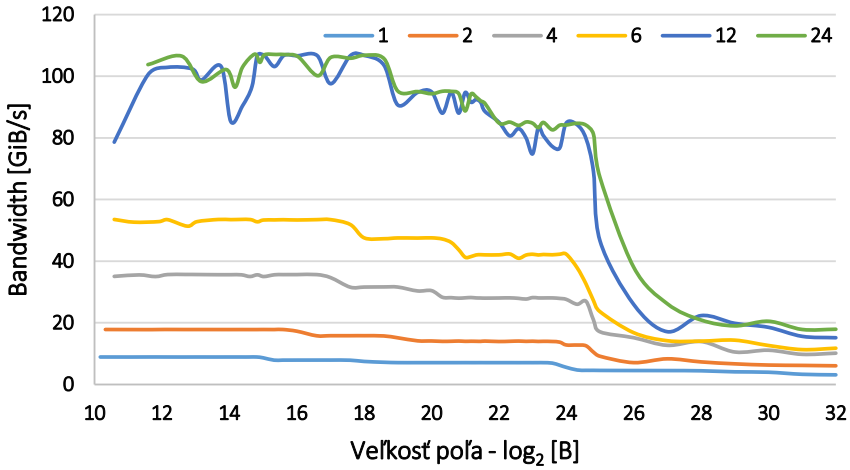
Obr. 5.12: Quicksort – priebeh spotreby CPU a RAM pre rôzny počet vlákien, nastavenie frekvencie CPU na 2400 MHz.



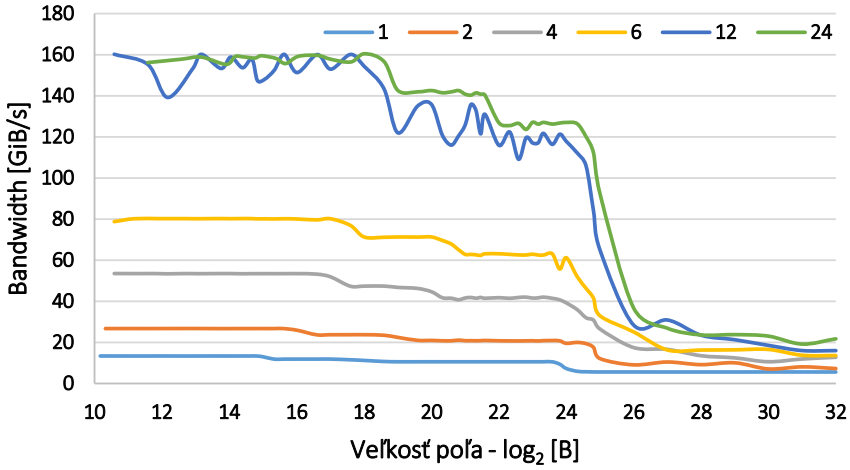
Obr. 5.13: Optimalizácia – násobenie matíc (1800 MHz).



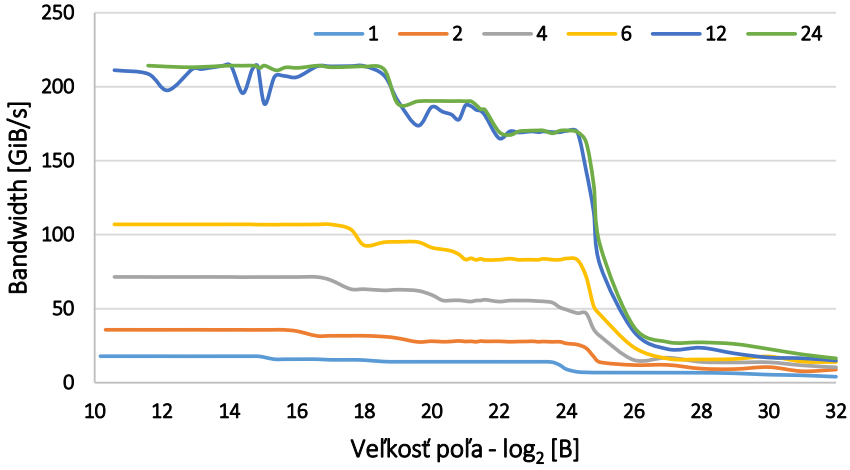
Obr. 5.14: Optimalizácia – quicksort (1200 MHz).



(a) Frekvencia CPU – 1200 MHz

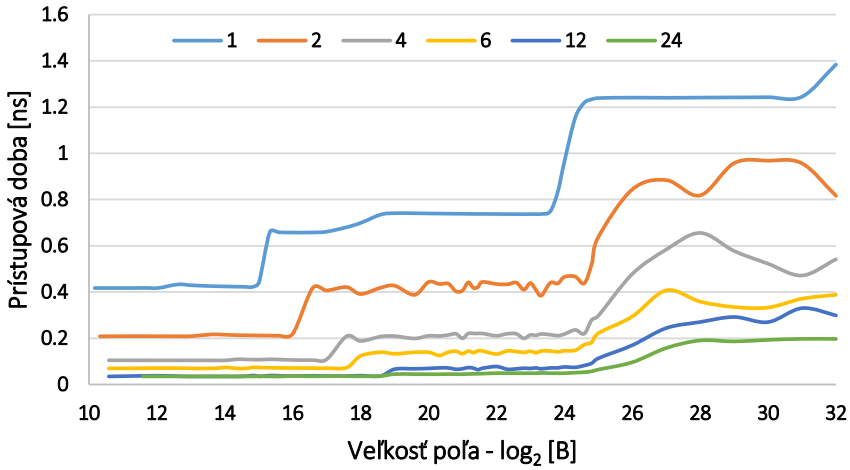


(b) Frekvencia CPU – 1800 MHz

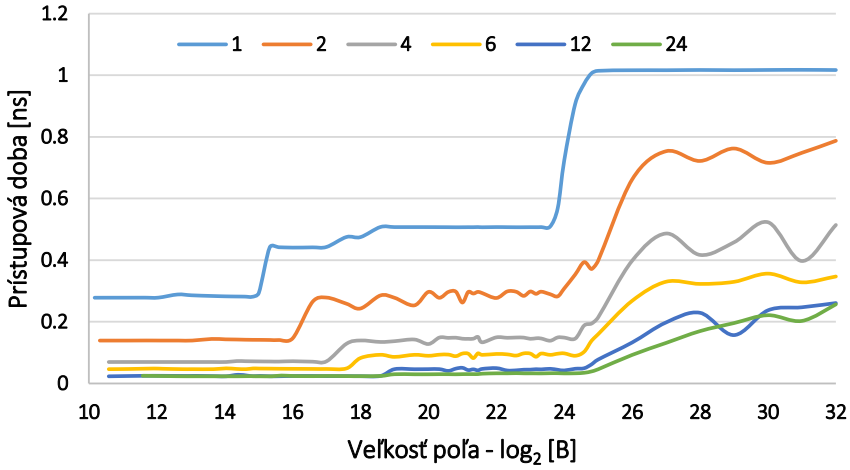


(c) Frekvencia CPU – 2400 MHz

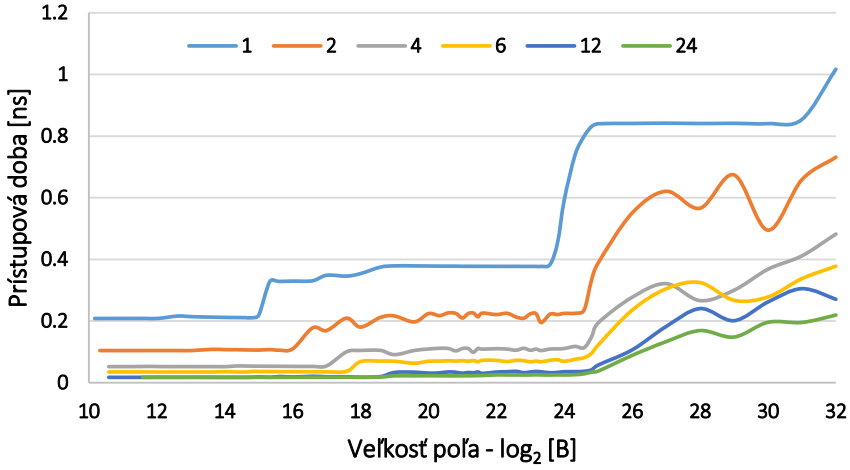
Obr. 5.15: PMBW – priepustnosť pamäti pri rozdielnom počte vlákien a rôznych frekvenciách (zápis).



(a) Frekvencia CPU – 1200 MHz

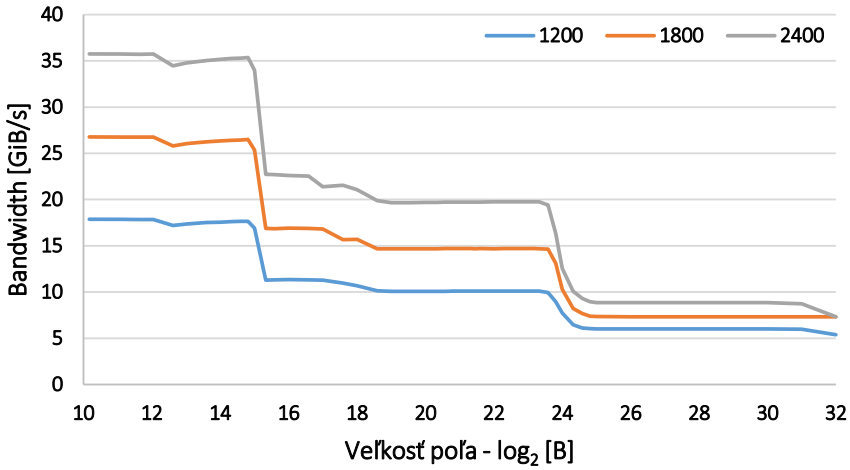


(b) Frekvencia CPU – 1800 MHz

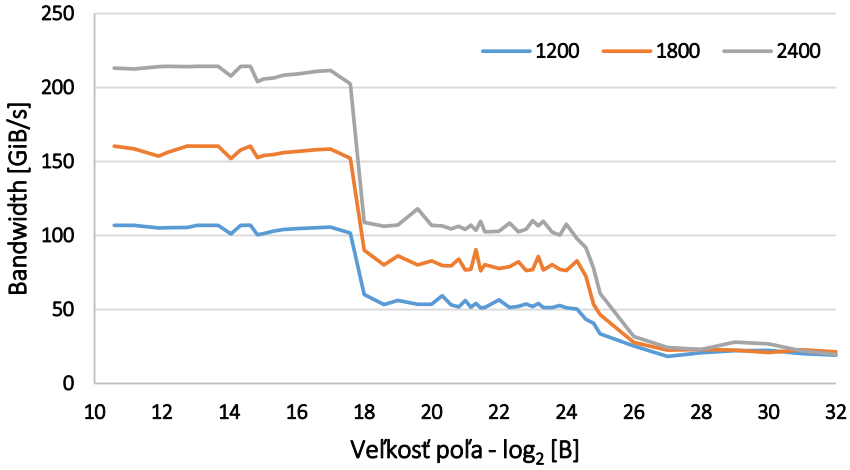


(c) Frekvencia CPU – 2400 MHz

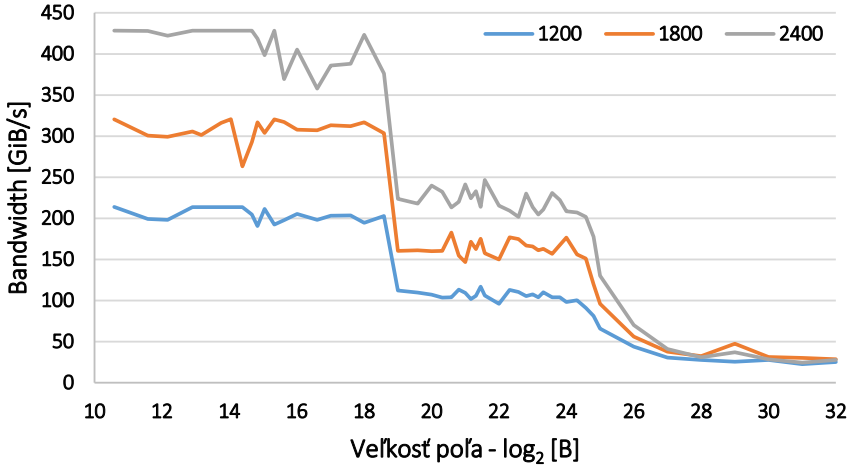
Obr. 5.16: PMBW – oneskorenie pamäti pri rozdielnom počte vlákien a rôznych frekvenciách (čítanie).



(a) IndexUnroll64 – 1 vlákno

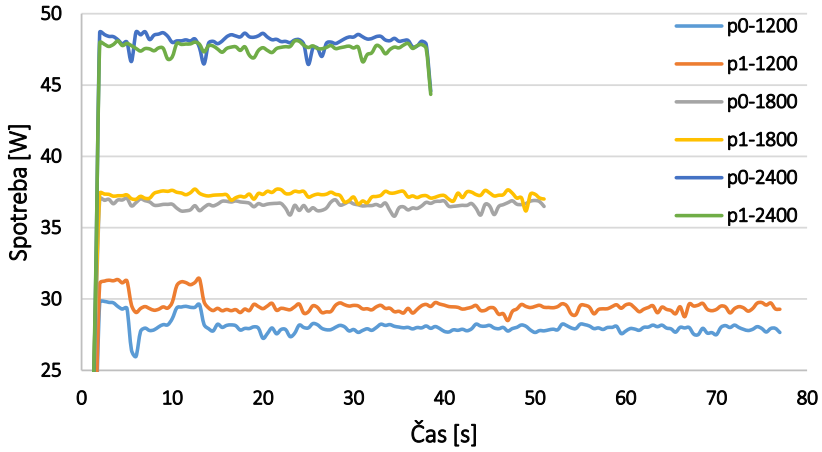


(b) IndexUnroll64 – 6 vlákien

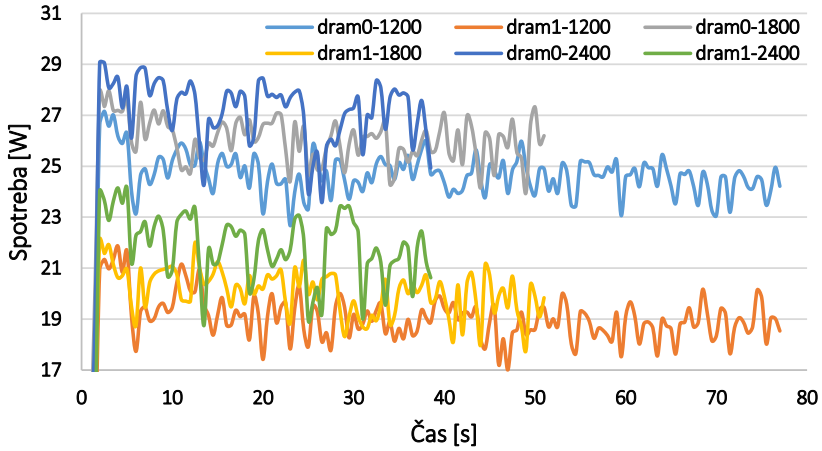


(c) IndexUnroll64 – 12 vlákien

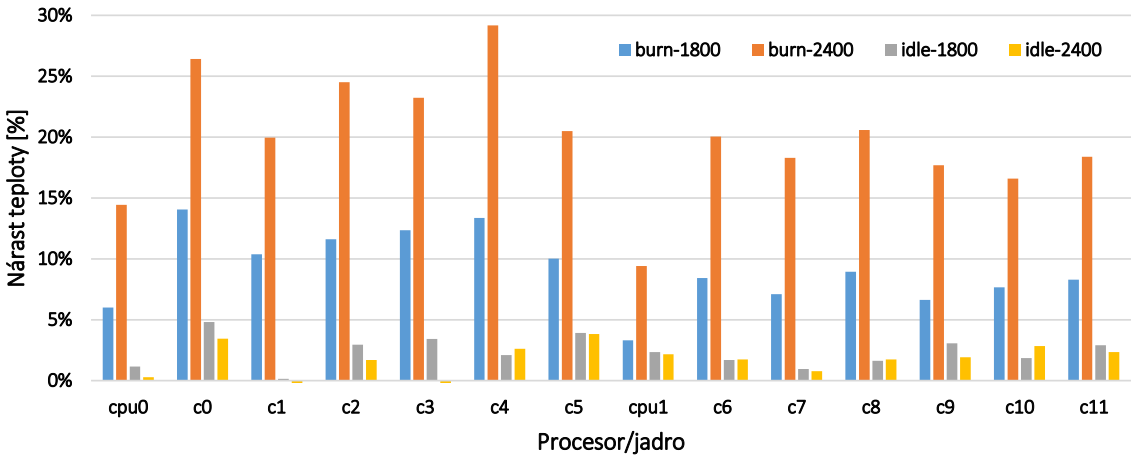
Obr. 5.17: PMBW – priepustnosť pamäti v závislosti od frekvencie CPU, pre rozdielny počet vlákien 1, 6, 12 (čítanie).



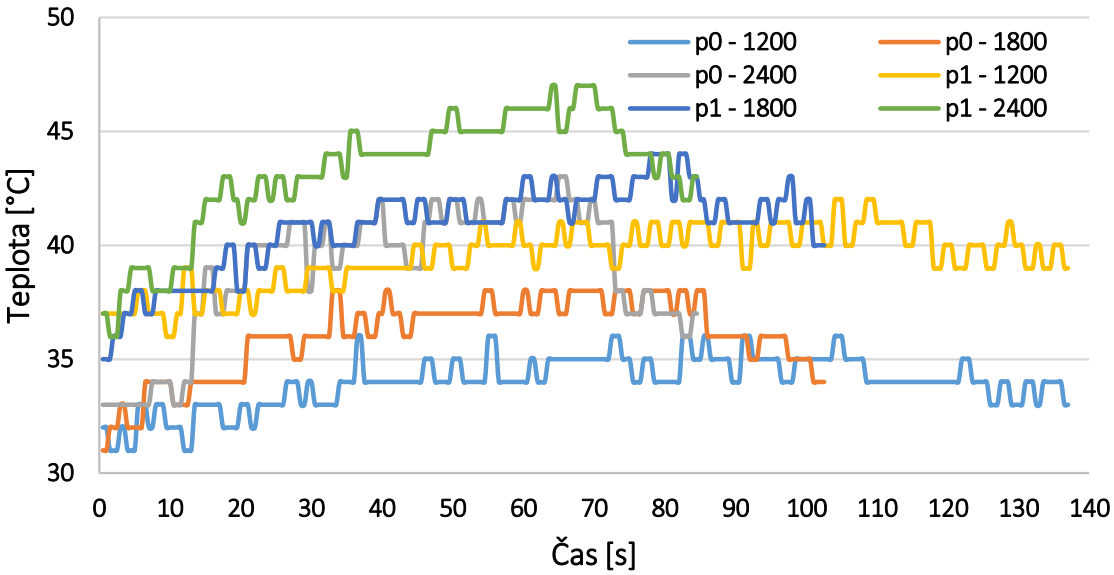
Obr. 5.18: Priebeh spotreby procesorov pri rôznych frekvenciách – LAMMPS SPC/E.



Obr. 5.19: Priebeh spotreby pamäti pri rôznych frekvenciách – LAMMPS SPC/E.



Obr. 5.20: Percentuálny nárast teploty pri vyšších frekvenciách v porovnaní s najnižšou (1200 MHz) – Linpack.



Obr. 5.21: Priebeh teploty procesorov a jadier počas výpočtu Linpack.

Kapitola 6

Záver

Cieľom tejto práce bolo zistiť, že do akej miery ovplyvňuje frekvencia čas výpočtu, spotrebu energie a teplotu procesora. Skúmanie tohoto vplyvu prebiehalo na navrhnutej sade benchmarkov, ktorá pozostáva z jednovláknových a paralelných výpočtových problémov. Testovanie prebiehalo na troch rôznych nastaveniach frekvencie (1200/1800/2400 MHz) procesoru.

V teoretickej časti práce sú riešené problémy z viacerých oblastí. Prvá oblasť je zameraná na procesory Intel. Táto časť obsahuje všeobecný popis procesora a popis použitej architektúry. Pojednáva aj o možnostiach zmeny frekvencie CPU, merania jeho spotreby, teploty a výkonu. Predmetom druhej oblasti je rozdelenie paralelných architektúr, na základe spôsobu organizácie pamäte, a popis modelov paralelného programovania.

Ďalej sú popísané benchmarky použité v experimentálnej časti práce. Jej obsahom je popis testovaného hardvéru, spôsobu testovania, namerané výsledky a ich zhodnotenie. Zaznamenané údaje sú zobrazené v tabuľkách a grafoch.

Zo získaných výsledkov je možné usúdiť, že frekvencia procesora vplýva na spotrebovanú energiu. Tento vplyv je závislí od druhu výpočtového problému. „Čistá“ spotreba energie jednovláknových benchmarkov sa zvýšila priemerne o 69 %, pri použití najvyššej frekvencií procesora. Testy viacvláknových benchmarkov ukázali, že pre výpočtový problém `násobenie matíc`, je vhodnejšie použitie najvyššej frekvencie. Priemerná spotreba zo všetkých spustení tohoto benchmarku dosahovala rovnakých až nižších hodnôt, ako pri teste na najnižšej frekvencií. Benchmark `quicksort` spotreboval za rovnakých okolností v priemere o 22 % viac energie. Pri všetkých testoch bola dosiahnutá 50% akcelerácia času, v porovnaní s najmenšou frekvenciou. Vzhľadom k celkovej spotrebe energie je však najvýhodnejšie použitie najvyššej frekvencie CPU.

Spotreba a čas výpočtu môže byť do veľkej miery ovplyvnený aj spôsobom prekladu programu. V práci sú porovnané optimalizačné sady `O0, O2, O3`. Najlepšie výsledky boli dosiahnuté použitím optimalizácie `O3`, pri benchmarku `quicksort`. Najväčšie zaznamenané zlepšenie spotreby energie a času výpočtu činí 70 %.

Náplňou práce bola tiež analýza vplyvu frekvencie procesora na priepustnosť pamäte. So získaných údajov vyplýva, že priepustnosť rýchlej vyrovnávacej pamäte cache je lineárne závislá na frekvencií procesora. Na rýchlosť hlavnej pamäte má frekvencia minimálny vplyv.

Experimenty v tejto práci prebiehali iba na jednom počítači a na jednom type procesoru. V pokračovaní práce je možné vykonať testovanie na rôznych procesoroch, prípadne rozšíriť sadu benchmarkov o iné. Ďalším cieľom môže byť vykonanie podobnej analýzy v oblasti superpočítačov a HPC.

Literatúra

[1] *An Introduction to the Intel® QuickPath Interconnect*. Intel, 2009, [Online; cit. 28. 4. 2016].
URL http://www.intel.com/content/dam/doc/white-paper/quick-path-interconnect-introduction-paper.pdf

[2] *Monitor CPU Temperature with DTS and PECI*. Intel, 2010, [Online; cit. 22. 4. 2016].
URL http://www.intel.com/content/www/us/en/embedded/testing-and-validation/cpu-monitoring-dts-peci-paper.html#

[3] *Technical desktops and workstationswith Intel® Xeon® uncore technologies deliver the performance you need to innovate faster*. Intel, 2012, [Online; cit. 11. 4. 2016].
URL http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/xeon-e5-2600-1600-workstations-desktops-uncore-prod-brief.pdf

[4] *What is LAMMPS*. 2013, [Online; cit. 9. 2. 2016].
URL http://lammps.sandia.gov/doc/Section_intro.html

[5] *Intel® Xeon® Processor E5-2620 v3*. Intel, 2014, [Online; cit. 4. 2. 2016].
URL http://ark.intel.com/products/83352/Intel-Xeon-Processor-E5-2620-v3-15M-Cache-2_40-GHz

[6] *Introduction to Intel® Architecture*. Intel, 2014, [Online; cit. 5. 4. 2016].
URL http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ia-introduction-basics-paper.pdf

[7] *About the OpenMP ARB and OpenMP.org*. 2015, [Online; cit. 9. 2. 2016].
URL http://openmp.org/wp/about-openmp/

[8] *Intel® AVX-512 Instructions and Their Use in the Implementation of Math Functions*. Intel, 2015, [Online; cit. 27. 4. 2016].
URL http://arith22.gforge.inria.fr/slides/s1-cornea.pdf

[9] *Intel® Xeon® Processor E5-2600 v3 and Intel® C612 Chipset Brief*. Intel, 2015, [Online; cit. 4. 2. 2016].
URL http://www.intel.com/content/dam/www/public/us/en/documents/platform-briefs/xeon-e5-2600-v3-c612-chipset-brief.pdf

[10] *Intel® Xeon® Processor E5-2600 v3 Product Family with Intel® C612 Chipset*. Intel, 2015, [Online; cit. 2. 2. 2016].
URL http://www.intel.com/content/www/us/en/embedded/products/grantley/specifications.html

[11] *The Message Passing Interface (MPI) standard*. 2015, [Online; cit. 10. 4. 2016].
URL http://www.mcs.anl.gov/research/projects/mpi/

[12] *PAPI Overview*. 2016, [Online; cit. 3. 2. 2016].
URL http://icl.cs.utk.edu/papi/overview/index.html

[13] BACH, M.: *Intel CPUs: Xeon E5 vs. Core i7*. Puget Systems, 2015, [Online; cit. 2. 2. 2016].
URL https://www.pugetsystems.com/labs/articles/Intel-CPUs-Xeon-E5-vs-Core-i7-634

[14] BINGMANN, T.: *PMBW – Parallel Memory Bandwidth Benchmark/Measurement*. 2013, [Online; cit. 4. 2. 2016].
URL https://panthema.net/2013/pmbw

[15] COUNTS, T.: *RUNNING AVERAGE POWER LIMIT*. 2014, [Online; cit. 4. 1. 2016].
URL https://01.org/blogs/tlcounts/2014/running-average-power-limit-%E2%80%93-rapl

[16] GHOSH, M.: *Findings by Google on NUMA Performance*. Jún 2013, [Online; cit. 28. 4. 2016].
URL https://moinakg.wordpress.com/2013/06/05/findings-by-google-on-numa-performance/

[17] HIRVISALO, V.: *A short OpenMP tutorial*. Marec 2015, [Online; cit. 21. 4. 2016].
URL https://wiki.aalto.fi/display/t1065450/openmp+tutorial+2015

[18] HUDÁK, S.: *Vizualizácia Dát z Konfokálneho Mikroskopu*. Diplomová práca, Fakulta matematiky, fyziky a informatiky Univerzita Komenského, Bratislava, 2006, [cit. 3 .4. 2016].

[19] HUTCHESON, A.; NATOLI, V.: *Memory Bound vs. Compute Bound: A Quantitative Study of Cache and Memory Bandwidth in High Performance Applications*. 2011, [Online; cit. 6. 4. 2016].
URL http://stoneridgetechnology.com/wp-content/uploads/2014/12/ComputevsMemory.pdf

[20] JOHNSON, M.: *Processor Architectures*. Stanford, 2008, [Online; cit. 2. 2. 2016].
URL http://dragonbook.stanford.edu/lecture-notes/Stanford-CS143/18-Processor-Architectures.pdf

[21] KAMINSKY, A.: *Parallel Computer Architectures*. Marec 2007, [Online; cit. 28. 4. 2016].
URL https://www.cs.rit.edu/~ark/lectures/pj04/notes.shtml

[22] KENNEDY, P.: *Inside Intel's Xeon E5-2600 V3: Huge Performance Leaps*. Toms IT Pro, 2014, [Online; cit. 2. 2. 2016].
URL http://www.tomsitpro.com/articles/intel-xeon-e5-2600-v3-cpu-grantley-deep-dive,1-2167.html

[23] MANCHANDA, N.; ANAND, K.: *Non-Uniform Memory Access (NUMA)*. [Online; cit. 28. 4. 2016].
URL http://cs.nyu.edu/~lerner/spring10/projects/NUMA.pdf

[24] MUJTABA, H.: *Tick-Tock Development Model*. 2015, [Online; cit. 2. 2. 2016].
URL http://wccftech.com/intel-unleashes-haswell-ex-xeon-e7-v3-processors-18-cores-45-mb-l3%2Dcache%2D12-tb-ddr4-memory-support-57-billion-transistors

[25] MUTNURY, B.; PAGLIA, F.; MOBLEY, J.; aj.: *QuickPath Interconnect (QPI) design and analysis in high speed servers*. In *Electrical Performance of Electronic Packaging and Systems (EPEPS), 2010 IEEE 19th Conference on*, Round Rock, TX: IEEE Publishing, Október 2010, ISBN 9781424468652, s. 265–268, doi:10.1109/EPEPS.2010.5642789, [cit. 25. 4. 2016].

[26] RAJPUT, V.; KUMAR, S.; PATLE, V.: Performance Analysis of UMA and NUMA Models. *International Journal of Computer Science Engineering and Technology (IJCSET)*, ročník 2012, č. 2, 2012, ISSN 2231-0711, [cit. 28. 4. 2016].

[27] SCHONE, R.; HACKENBERG, D.; MOLKA, D.: *Memory Performance at Reduced CPU Clock Speeds: An Analysis of Current x86_64 Processors*. In *Presented as part of the 2012 Workshop on Power-Aware Computing and Systems*, Berkeley, CA: USENIX, 2012, [cit. 11. 5. 2016].
URL https://www.usenix.org/conference/hotpower12/workshop-program/presentation/Schone

[28] SEKANINA, L.: *Materiály predmetu INP*. FIT VUT v Brně, 2014, [Online; cit. 28. 4. 2016].

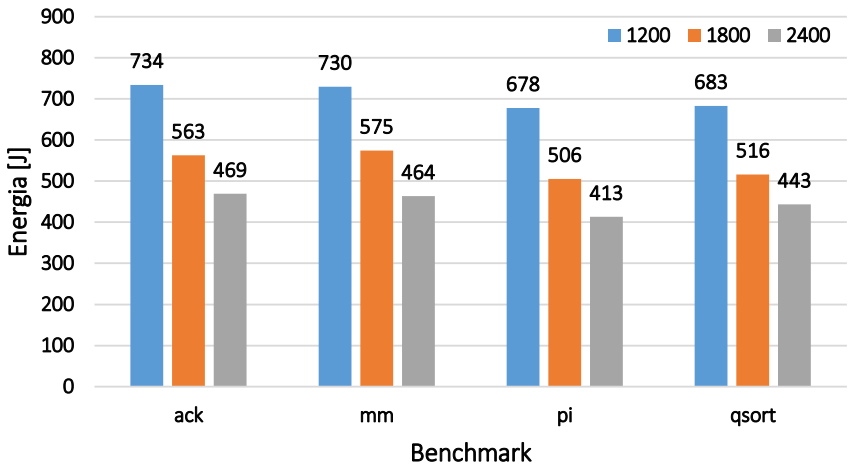
[29] TUDOR, B. M.; TEO, Y. M.: *A Practical Approach for Performance Analysis of Shared-Memory Programs*. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, Singapore: IEEE Publishing, 2011, ISBN 9781612843728, ISSN 15302075, s. 652–663, doi:10.1109/IPDPS.2011.68, [cit. 4. 5. 2016].

Prílohy

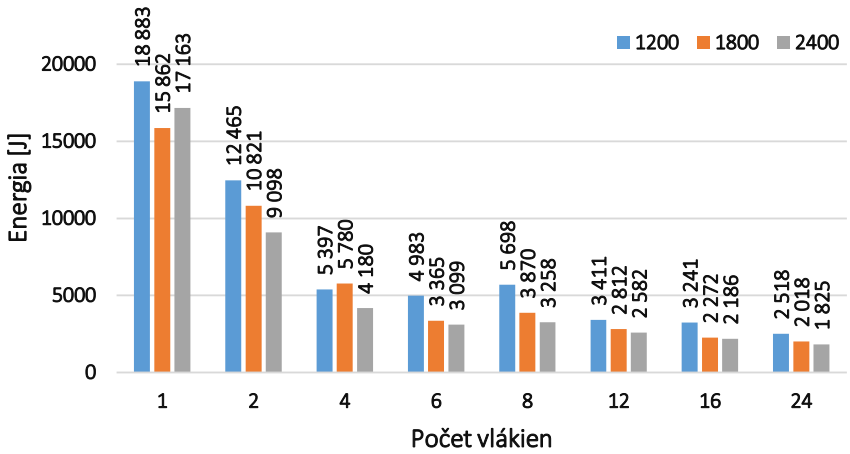
Zoznam príloh

Príloha A

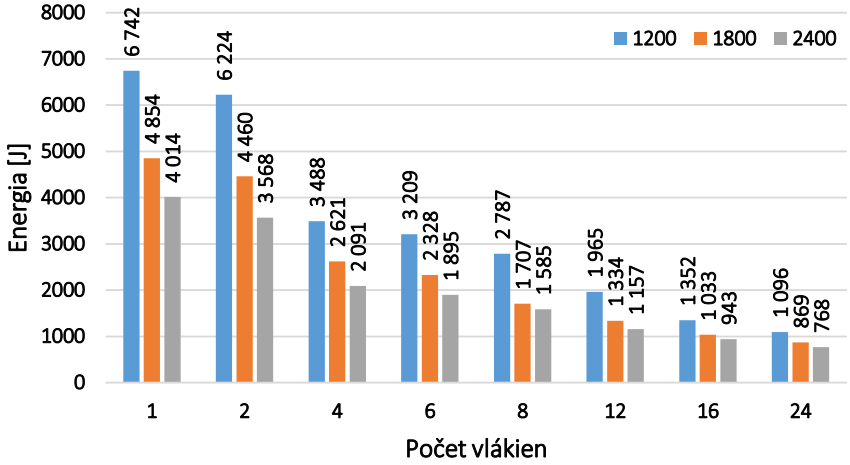
Grafy



(a) Jednovláknové benchmarky

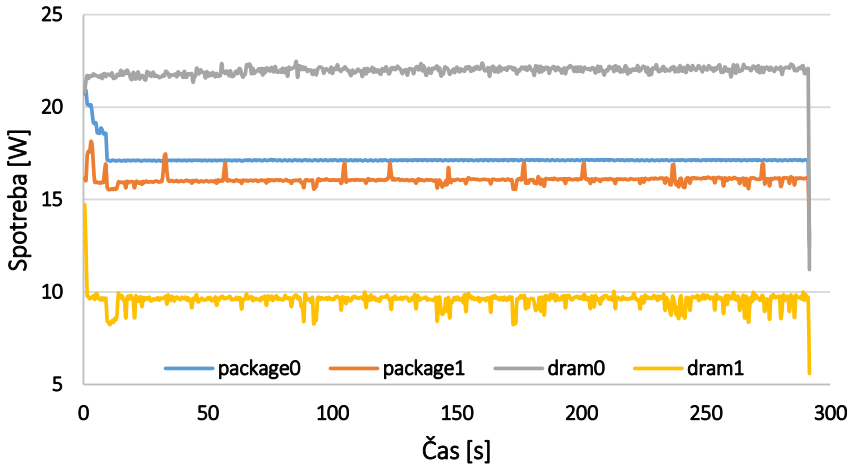


(b) Paralelné násobenie matíc

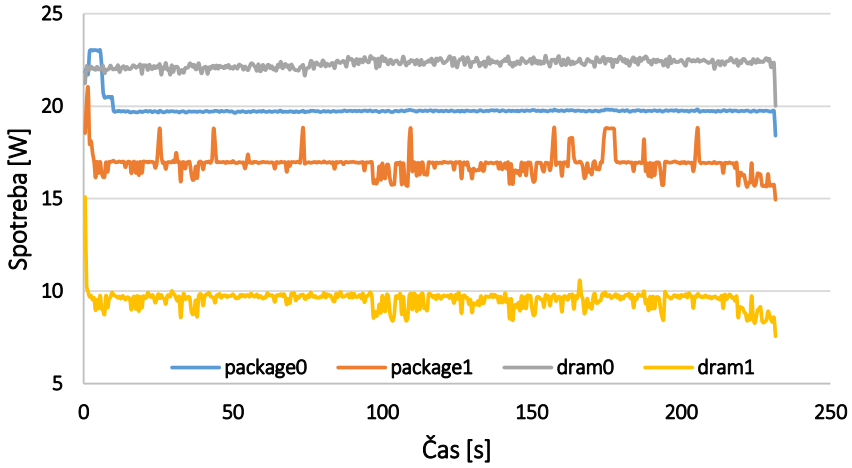


(c) Quicksort

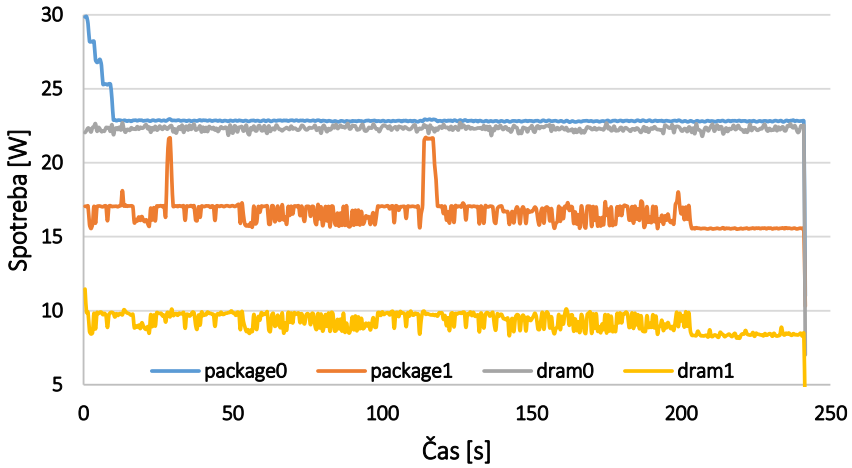
Obr. A.1: Celková spotreba energie CPU a RAM potrebná pre výpočet, pri nastavení CPU na rôzne frekvencie.



(a) Frekvencia CPU – 1200 MHz

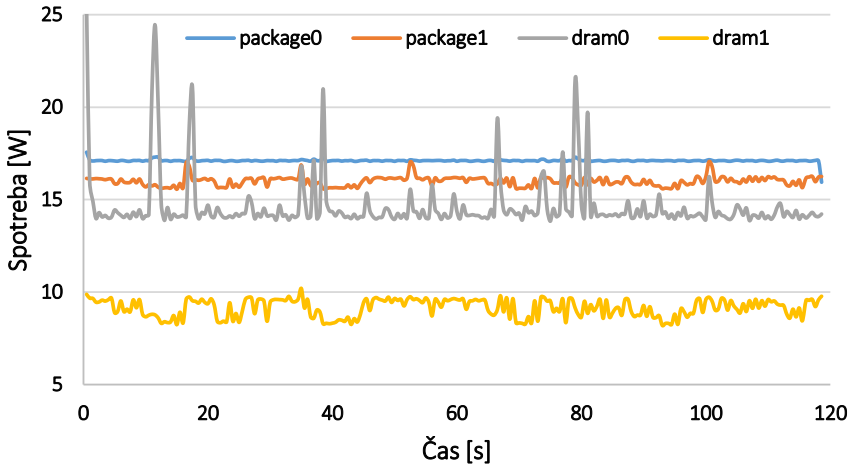


(b) Frekvencia CPU – 1800 MHz

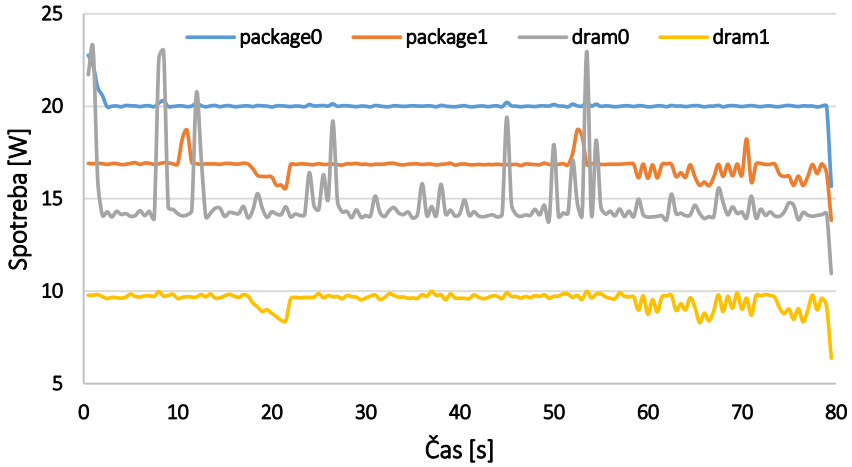


(c) Frekvencia CPU – 2400 MHz

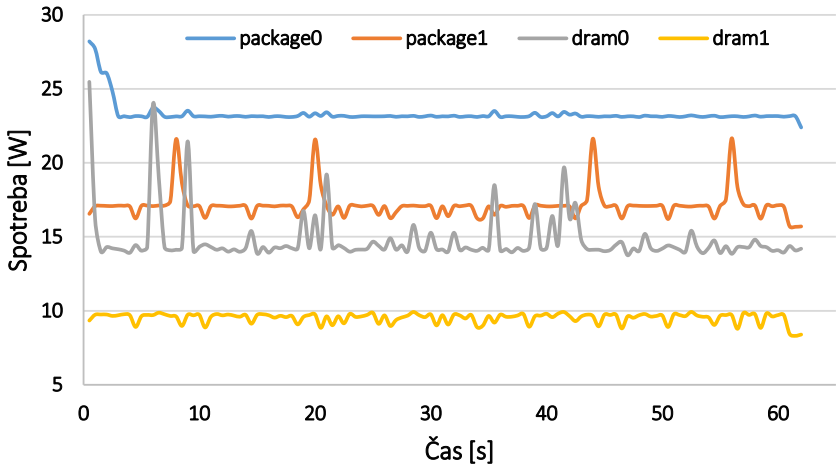
Obr. A.2: Paralelné násobenie matíc – priebeh spotreby procesorov a pamätí pre rôzne frekvencie, 1 vlákno.



(a) Frekvencia CPU – 1200 MHz

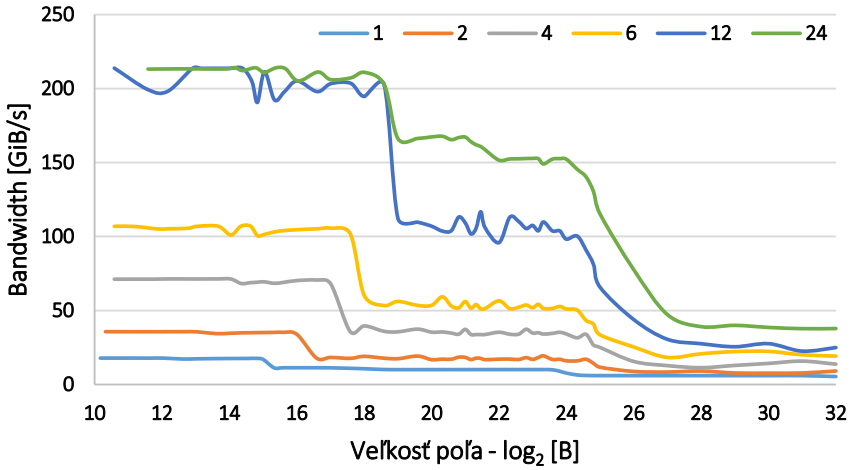


(b) Frekvencia CPU – 1800 MHz

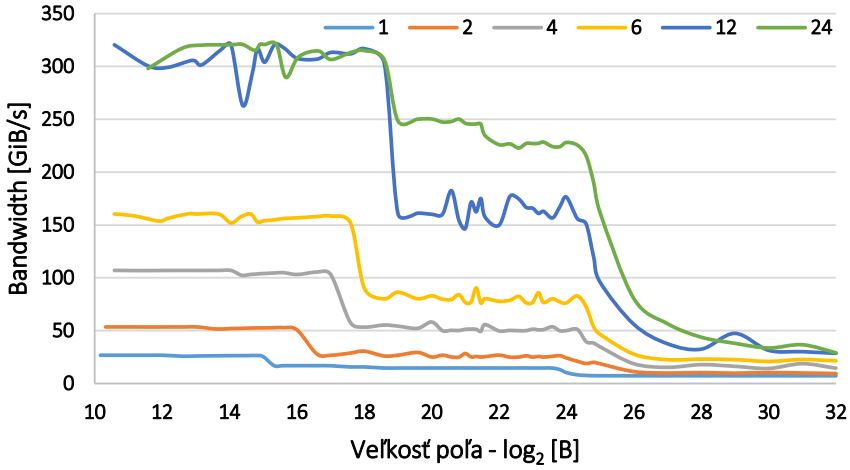


(c) Frekvencia CPU – 2400 MHz

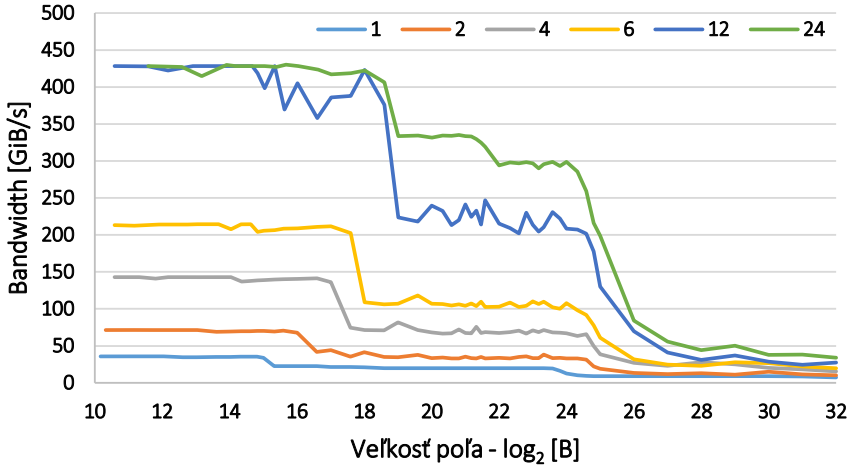
Obr. A.3: Quicksort – priebeh spotreby procesorov a pamätí pre rôzne frekvencie, 1 vlákno.



(a) Frekvencia CPU – 1200 MHz

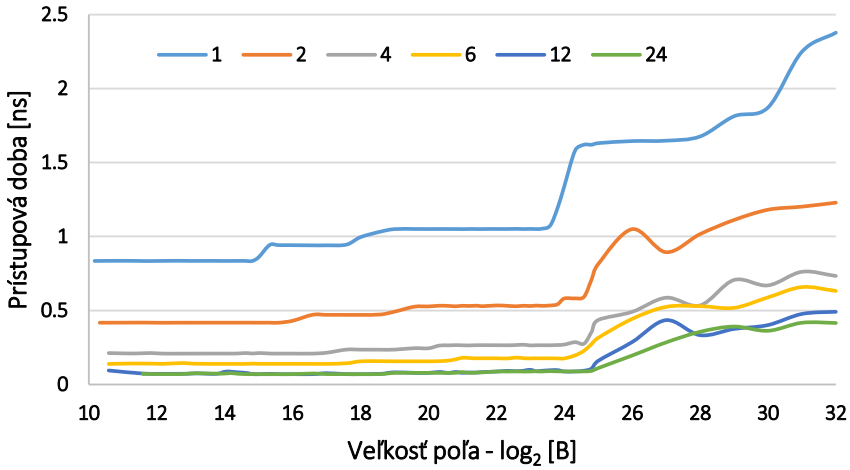


(b) Frekvencia CPU – 1800 MHz

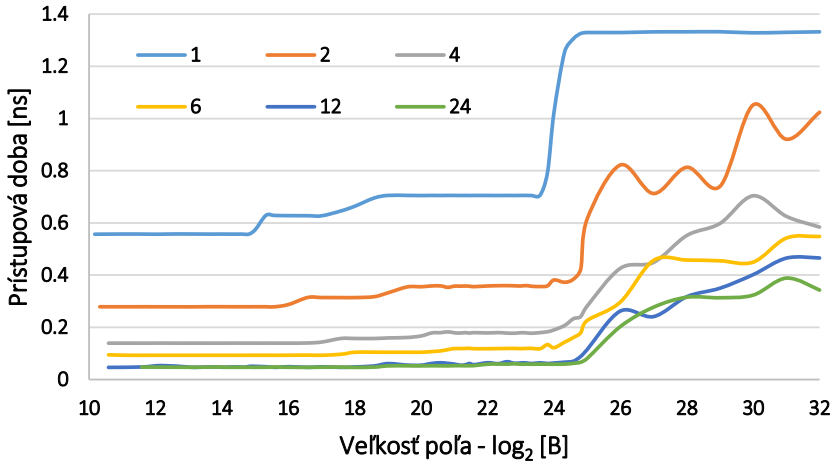


(c) Frekvencia CPU – 2400 MHz

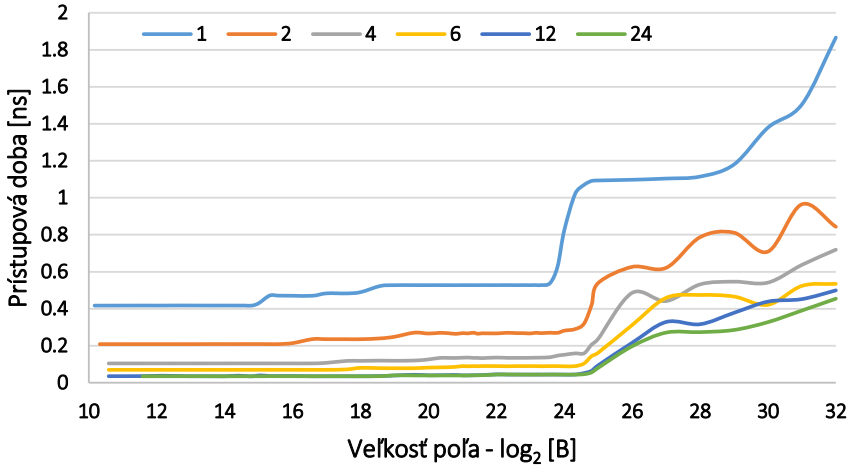
Obr. A.4: PMBW – priepustnosť pamäti pri rozdielnom počte vlákien a rôznych frekvenciách (čítanie).



(a) Frekvencia CPU – 1200 MHz

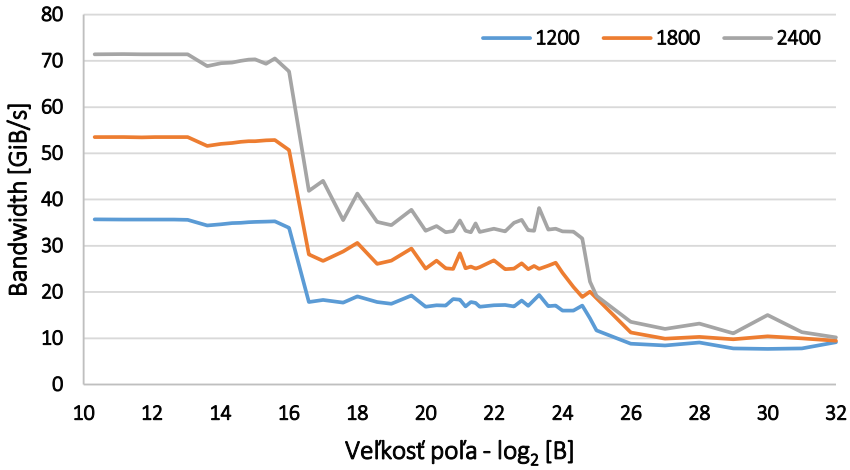


(b) Frekvencia CPU – 1800 MHz

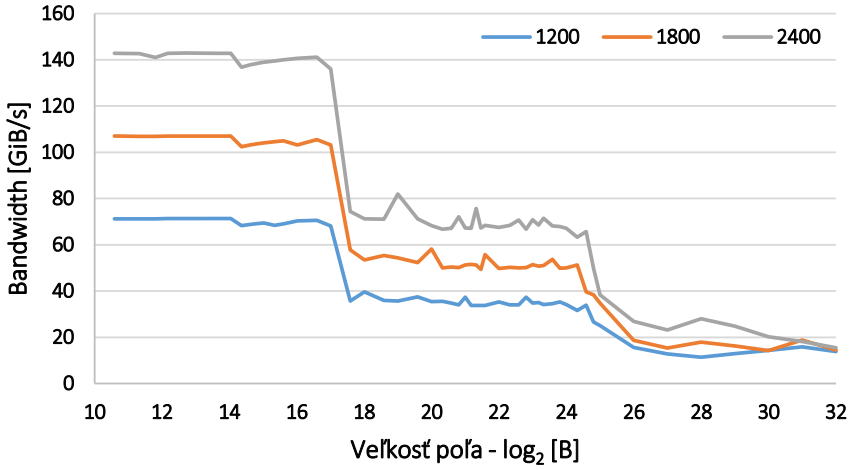


(c) Frekvencia CPU – 2400 MHz

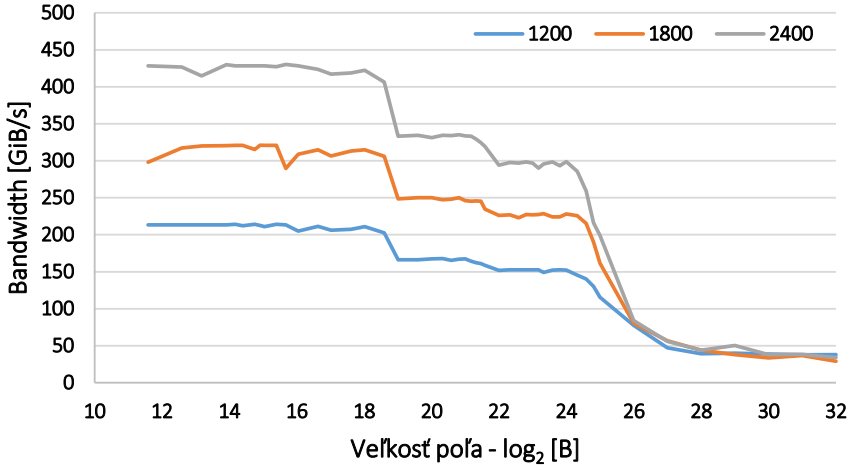
Obr. A.5: PMBW – oneskorenie pamäti pri rozdielnom počte vlákien a rôznych frekvenciách (zápis).



(a) IndexUnroll64 – 2 vlákna



(b) IndexUnroll64 – 4 vlákna



(c) IndexUnroll64 – 24 vlákien

Obr. A.6: PMBW – priepustnosť pamäti v závislosti od frekvencie CPU, pre rozdielny počet vlákien 2, 4, 24 (čítanie).

Príloha B

Tabuľky

f [MHz]	Vlákien	Čas [s]	Package0 [J]	Package1 [J]	Dram0 [J]	Dram1 [J]	Celkom [J]	Celkom [W]
1200	1	23.6	35.71	15.48	132.23	14.40	**197.82**	8.39
1200	2	11.8	43.25	8.23	79.84	7.14	138.46	11.70
1200	4	6.0	52.38	12.37	46.97	10.01	121.73	20.41
1200	6	4.0	52.58	7.80	34.55	4.84	99.78	24.90
1200	8	3.0	38.87	19.26	21.95	21.02	101.11	33.36
1200	12	2.1	28.20	30.88	13.99	16.19	89.25	42.65
1200	16	2.4	25.69	24.89	18.46	11.15	80.20	34.04
1200	24	2.3	26.85	29.36	18.84	11.77	86.82	38.37
1800	1	15.9	44.18	3.62	105.34	5.83	158.97	9.98
1800	2	8.0	56.97	10.67	61.64	9.97	139.25	17.39
1800	4	4.0	54.54	8.12	33.33	4.79	100.78	24.92
1800	6	2.7	50.68	1.14	21.93	1.35	75.11	27.79
1800	8	2.0	44.13	21.58	17.83	17.10	100.65	49.26
1800	12	1.4	22.70	24.77	10.21	5.23	**62.91**	44.37
1800	16	1.8	26.06	25.01	9.83	10.95	71.86	40.04
1800	24	1.5	29.71	33.01	12.10	12.39	87.21	57.09
2400	1	11.9	58.28	-4.59	71.77	0.06	125.52	10.52
2400	2	6.0	70.89	5.21	46.65	6.63	129.38	21.61
2400	4	3.0	59.88	4.07	25.39	4.20	93.54	30.91
2400	6	2.0	60.62	4.66	21.15	3.87	90.31	44.47
2400	8	1.5	46.26	22.40	16.65	9.50	94.80	61.78
2400	12	1.1	31.17	33.54	12.74	7.64	85.08	78.95
2400	16	1.3	32.02	26.69	8.69	8.03	75.43	57.66
2400	24	1.2	30.80	33.35	11.81	6.84	82.81	71.62

Tabuľka B.1: Paralelné násobenie matíc (veľkosť 1024^2) – spotrebovaná energia pri rôznom nastavení.

f [MHz]	Vlákien	Čas O3 [s]	Čas O2 [s]	Čas O0 [s]	Celkom O3 [J]	Celkom O2 [J]	Celkom O0 [J]
1200	1	38.5	39.5	118.2	153.69	126.52	**217.24**
1200	2	35.3	36.1	107.9	170.68	125.68	267.08
1200	4	18.7	18.7	58.0	151.67	135.65	284.38
1200	6	14.0	14.2	52.4	146.01	139.35	315.87
1200	8	11.0	10.1	42.8	138.67	201.08	423.77
1200	12	8.5	7.5	28.1	212.12	193.79	412.68
1200	16	7.0	6.2	17.3	194.40	210.73	394.42
1200	24	4.4	4.1	12.4	171.24	155.28	408.63
1800	1	25.7	26.1	78.9	143.67	130.61	315.73
1800	2	23.5	23.1	72.0	141.22	133.42	315.57
1800	4	12.7	12.3	39.7	**121.38**	**122.84**	335.18
1800	6	9.9	9.5	34.8	153.68	145.20	324.29
1800	8	6.8	7.3	22.0	199.06	212.47	442.57
1800	12	6.3	6.3	15.7	165.14	211.22	428.29
1800	16	4.2	4.6	10.8	144.42	173.49	413.29
1800	24	2.8	3.0	8.4	170.87	131.90	382.37
2400	1	19.3	20.8	61.5	170.38	185.82	371.31
2400	2	17.8	18.1	53.9	145.29	151.83	375.04
2400	4	9.4	10.3	29.1	187.64	191.38	370.98
2400	6	7.3	7.1	25.9	187.36	159.45	364.25
2400	8	7.5	5.5	18.4	210.77	199.34	493.90
2400	12	3.7	3.7	11.9	166.31	165.97	451.86
2400	16	3.3	3.5	9.1	196.90	151.35	401.69
2400	24	**2.3**	**2.2**	**6.4**	170.81	145.63	387.65

Tabuľka B.2: Porovnanie optimalizácie z hľadiska času a spotreby energie – quicksort.

f [MHz]	Vlákien	Čas O3 [s]	Čas O2 [s]	Čas O0 [s]	Celkom O3 [J]	Celkom O2 [J]	Celkom O0 [J]
1200	1	269.2	262.8	290.6	2660.92	2452.25	2839.44
1200	2	55.6	42.6	188.0	795.21	681.39	2085.34
1200	4	30.3	30.9	76.2	579.57	540.03	1192.47
1200	6	26.3	26.6	68.8	531.15	**516.44**	1186.08
1200	8	40.9	22.9	71.8	1164.21	721.08	1735.91
1200	12	23.5	23.6	37.8	971.08	980.22	1322.51
1200	16	26.2	25.7	35.8	1010.62	1031.19	1267.38
1200	24	24.3	24.6	27.2	974.21	968.06	1014.99
1800	1	182.1	188.0	231.0	1879.27	2228.98	2564.04
1800	2	41.4	74.5	152.6	774.80	1146.65	2040.41
1800	4	29.0	28.9	75.3	684.05	698.84	1446.23
1800	6	19.7	20.7	41.70	**512.57**	523.15	963.14
1800	8	19.9	24.2	42.3	843.17	900.68	1433.44
1800	12	16.3	13.1	26.9	897.84	602.09	1264.74
1800	16	18.3	15.4	20.9	936.76	882.56	1066.18
1800	24	16.6	12.9	18.9	888.55	927.49	**929.88**
2400	1	137.4	47.2	241.0	1782.83	872.26	2898.50
2400	2	68.7	68.4	120.6	1411.84	1385.72	1961.21
2400	4	34.5	35.1	51.0	940.31	1053.81	1159.91
2400	6	21.3	23.3	35.0	738.21	818.14	1027.25
2400	8	18.4	28.2	30.9	911.28	1185.53	1427.52
2400	12	**9.8**	**12.0**	21.8	652.95	897.33	1288.62
2400	16	11.9	13.6	17.5	830.91	917.89	1150.48
2400	24	11.7	12.6	**14.9**	801.63	881.50	945.03

Tabuľka B.3: Porovnanie optimalizácie z hľadiska času a spotreby energie – násobenie matíc

Príloha C

Obsah CD

- **`benchmarky/`** – Obsahuje priečinky každého benchmarku a súbor s celkovým zhrnutím výsledkov vo formáte `.xslx`
  - **`jednovlaknove_bench/`** – Výsledky s benchmarkov a zdrojové súbory
  - **`lammps/`** – Použité vstupné skripty a zdrojové súbory
  - **`linpack/`** – Binárne súbory benchmarku
  - **`nasobenie_matic/`** – Výsledky a zdrojové súbory
  - **`meranie/`** – Výsledky merania teploty a spotreby v idle, zdrojové súbory
  - **`pmbw/`** – Binárne súbory benchmarku pre OS Linux a Windows
  - **`quicksort/`** – Výsledky a zdrojové súbory
  - **`zhrnutie.xlsx`** – Porovnanie výsledkov pretaktovania
- **`projekt_text/`** – Zdrojové súbory technickej správy
- **`projekt.pdf`** – Technická správa
- **`README`** – Súbor s nápovedou, popisuje štruktúru súborov na CD