TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N3901 – Aplikované vědy v inženýrství
Studijní obor: 3901T025 – Přírodovědné inženýrství

Paralelní výpočty proudění

Parallel CFD simulations

Diplomová práce

Autor: **Bc. Václav Řidký**

Vedoucí práce: Ing. Petr Šidlof, Ph.D

Konzultant: Mgr. Jan Březina, Ph.D.

V Liberci 20. 5. 2011

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem

Datum: 20.5.2011

Podpis:

Poděkování

Chtěl bych poděkovat hlavně vedoucímu této diplomové práce Ing. Petru Šidlofovi Ph.D, konzultantovi pro problematiku paralelního počítaní Mgr. Janu Březinovi, Ph.D. a také i dalším lidem kteří mi s touto prací pomohli. Jejich rady a zkušenosti mi pomohly ke vzniku této práce. Zároveň také děkuji své rodině za poskytnutou trpělivost a zázemí.

Abstrakt

V diplomové práci je řešena problematika CFD simulací od sestrojení geometrie řešené oblasti, její následné zasíťování, až po provedení samotného výpočtu a ověření správnosti výsledku. Hlavní důraz pak je kladen na výpočetní část, a to hlavně na možnost paralelizace výpočtu a s tím spojenou problematiku. Hlavní parametr u paralelizace je efektivnost a zrychlení paralelního výpočtu, proto byl výpočet proudového pole spouštěn na různém počtu jader a sleduje se výpočetní čas.

Na vymodelování geometrie úlohy a její zasíťovaní byl požit program GMSH. Jako výpočetní nástroj byl použit výpočetní balík OpenFOAM. Výpočetní oblastí je kontejner na testování účinnosti nanovlakených filtrů. Pro paralelní výpočty byl využit školní výpočetní cluster *Hydra*.

Klíčová slova: sítě, GMSH, OpenFOAM metoda konečných prvků a objemů, CFD výpočty

The aim of this thesis is to introduce the issue of CFD simulations by constructing geometry of the given domain, mesh generation, computing and verifying the accuracy of results. The main emphasis will be placed on the computational part and especially the possibility of parallelization and the associated problems. The main parameter for the parallelization is the efficiency of parallel computation. The CFD simulations were run on different numbers of processors and cores per node, and the computational time was monitored. The geometry and mesh generation was realized in the GMSH software, the simulations were run within the OpenFOAM software package. The computations were run in parallel on a the computational cluster *Hydra*

Keywords: mesh, GMSH, OpenFoam, finite element method and finite volume methods, CFD simulations

Obsah

Úvod

V této práci se spojují poznatky z několika vědních disciplin, jako je mechanika tekutin, numerická matematika a informatika. Ty se aplikují na konkrétní úlohu, kterou je v tomto případě prouděni v uzavřené nádobě. Tato nádoba slouží pro testování účinnosti nanovlákených filtračních materiálů.

Výsledky získané simulacemi budou použity pro další práci na této problematice. Část zabývající se paralelními výpočty, pak poslouží jako matematická podpora při použití filtrů v prostorově rozsáhlé oblasti. Řešení úlohy proudění v rozsáhlé oblasti za pomoci sekvenčního výpočtu je velice náročně (nároky na paměť) a doba výpočtu neúměrně dlouhá, jelikož simulace rozsáhlé oblasti budou vyžadovat sítě o mnoha milionech elementů. Příkladem rozsáhlé oblasti je čistění spalin v průmyslové výrobě a hlavně pak ve spalovně komunálního odpadu.

Simulace proudění se řeší numericky pomocí výpočetního balíku OpenFOAM. Jeho největší výhodou je, že se jedná open-sorce program s možností zásahu do výpočetního kódu, takže uživatel má možnost přizpůsobit výpočetní algoritmus pro svou problematiku. Další výhodou je, že spolupracuje s velkým množstvím jak veřejně tak komerčně používaných generátorů sítí a navíc je přímo propojen s vizualizačním prostředí paraview, které nabízí širokou paletu nástrojů na tvorbu vizualizací, a to i včetně animací.

Veškeré výpočty byly spouštěny na školním výpočetním clusteru HYDRA. Výpočetní cluster disponuje 78 jádry, což umožňuje rozsáhlou paralelizaci. Jedná se o smíšený (hybridní) cluster. Pro vyhodnocení paralelních výpočtů byla zhodnocena efektivnost a zrychlení paralelního výpočtu, a také porovnání výpočetní doby v závislosti na počtu a typu procesorů.

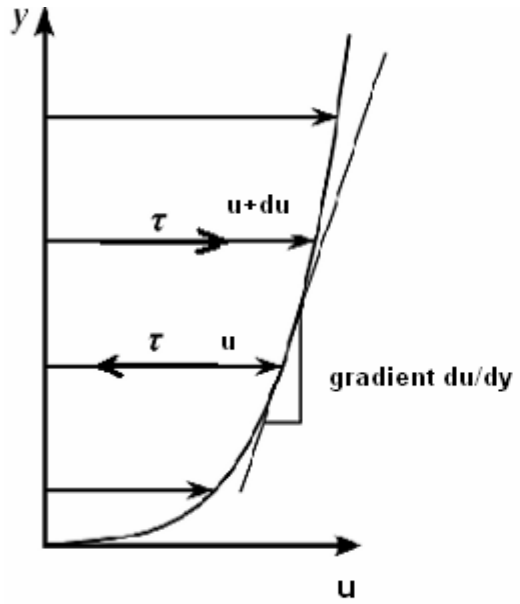
Práce byla částečně podpořena z projektu GAČR P101/11/0207 "Coupled problems of fluid and solid mechanics - nonlinear aeroelasticity".

2 Úvod do mechaniky tekutin

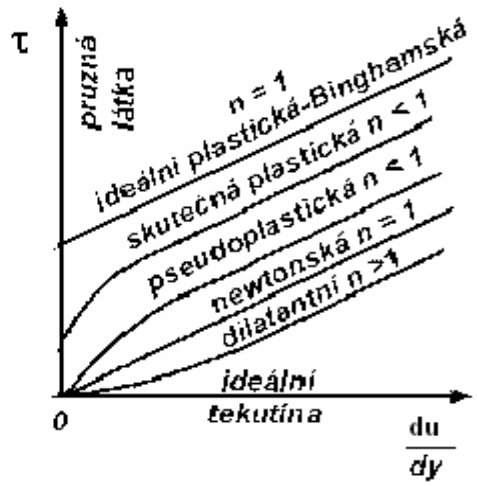
2.1 Vlastnosti tekutin

Tekutina se bere v klasické mechanice tekutin jako kontinuum z důvodu snazšího matematického popisu. Tekutiny se vyznačují tím, že nemají vlastní tvar, což je způsobeno malými mezimolekulárními silami. Tekutiny se dělí na kapaliny a plyny. Hlavní rozdíl mezi kapalinou a plynem je ten, že plyn vyplní celý objem uzavřené nádoby na rozdíl od kapaliny, která zaplní spodní část nádoby a ustaví se volná hladina. Další, v čem se kapaliny liší, jsou jejich vlastnosti, jako je například stlačitelnost, hustota a viskozita.[6]

Viskozita je vlastnost kapaliny, která způsobuje vnitřní tření kapaliny. Vnitřní tření vyvolá v tekutině třecí sílu mezi dvěmi vrstvami tekutiny, to způsobí vytvoření rychlostního profilu proudící tekutiny (viz obr.2.1). Viskozita tekutiny je dvojího druhu, a to dynamická viskozita a kinematická viskozita. Dynamická viskozita je materiálová vlastnost tekutiny a je závislá na teplotě. Pro plyny s rostoucí teplotou roste a pro kapaliny klesá. Kinematická viskozita je dána poměrem dynamické viskozity a hustoty, to má za následek, že u plynů je kinematická viskozita výrazně závislá na tlaku a také na teplotě. Všechny tyto závislosti jsou převážně nelineární, ale z pohledu modelování proudění se dají považovat převážně za lineární nebo dokonce konstantní, jelikož se pracuje většinou v malém rozsahu tlaku a teploty nebo při konstantní teplotě.[6]



Obr. 2.1: Vliv viskozity na rychlostní profil [4]



Obr. 2.2: Závislost tečného napětí na gradientu rychlosti [4]

Dle viskozity se kapaliny dělí na newtonovské, nenewtonovské a ideální. Ideální tekutina je tekutina bez vnitřního tření, smyková napětí jsou nulová, jedná se o zjednodušený případ. Někdy se používá název nevazké tekutiny namísto ideální. Tato idealizace se využívá zřídka, protože reálné kapaliny mají vnitřní tření. Newtonovské kapaliny se vyznačují lineární závislostí smykového napětí na gradientu rychlosti (viz obr.2.2). Vztah pro napětí je popsán Newtonovým zákonem viskozity

$$\tau = \eta \frac{du}{dy} \ , \qquad (2.1)$$

kde τ je smykové napětí, η je dynamická viskozita a *du/dy* je gradient rychlosti. Příkladem Newtonovské kapaliny je z kapalin voda a z plynných látek například vzduch.

Nenewtonovských kapalin je několik druhů, jsou rozlišeny dle závislostí tečného napětí na gradientu rychlosti, neplatí u nich Newtonův zákon viskozity. Jedná se především o taveniny plastů, barvy a suspenze. Nenewtonovské kapaliny se dělí na tři základní skupiny. Jedná se tekutiny Binghamské, pseudoplastické, dilatantní.

Binghamské tekutiny se vyznačují tím, že tekutina začne téct až po překročení prahového smykového napětí, do té doby se chová jako pevná látka. Mezi tyto kapaliny patří kaly a husté suspenze. Pseudoplastické tekutiny jsou tekutiny u nichž se s rostoucím gradientem rychlosti snižuje viskozita. Dilatantní jsou přesným opakem pseudoplastické tekutiny, protože s rostoucím gradientem rychlosti roste viskozita tekutiny.

Viskozita má velký význam při modelování proudění, jelikož výrazně ovlivňuje rychlostní pole. Čím je viskozita větší, tím je potřeba vyšší rychlost, aby proudění přešlo z laminárního na turbulentní. Této problematice bude věnována větší pozornost v kapitole zabývající se dynamikou tekutin. [4]

Další vlastností tekutiny je stlačitelnost. Stlačitelnost se projevuje hlavně u plynů, většina kapalin se bere jako nestlačitelné. Stlačitelnost je vlastnost tekutiny měnit objem při působení vnějšího tlaku. Plyny jsou velice stlačitelné dokonce může docházet k jejich stlačování i při samotném proudění. Mezní rychlost, pro kterou dochází ke stlačování plynu vlivem proudění je přibližně 0,3 násobek Machova čísla pro daný plyn. Pro nižší rychlosti se bere plyn jako nestlačitelný a hustota je tedy nezávislá na rychlosti. Jedná se o výrazné zjednodušení matematického modelu proudění plynu a tím i zrychlení výpočtu. Kapaliny jsou v tomto ohledu pro modelovaní prodění mnohem jednoduší, jelikož se berou jako nestlačitelné [6]

2.2 Kinematika tekutin

Kinematika tekutin se zabývá popisem proudící tekutiny bez ohledu na působení vnějších sil. K popisu proudící tekutiny se využívají dva přístupy. Těmito přístupy jsou Eulerův a Lagrangeův. Eulerův popis je v mechanice tekutin používán častěji, snáze se pak sestavuje matematický model odvozený z Eulerova popisu. Eulerův popis bere tekutiny jako kontinuum a zaměřuje se na tekutinu jako vnější pozorovatel. Popis je založen na pozorování v určitém bodu tekutiny a ne konkrétní částice, jako je tomu u Lagrangova popisu. V Eulerově popisu je rychlost funkcí prostoru a času. Při konstantním čase je rychlost funkcí prostoru a popisuje rychlostní pole. [8]

Kinematika definuje základní křivky pro snažší představu pohybu tekutiny v objemu. Z Lagrangeova popisu jednoduše získáme trajektorii, jedná se o křivku po níž se pohybuje určitá částice tekutiny. Eulerův popis je vhodný k určování proudnic. Proudnice jsou křivky, na které je v každém místě proudící tekutiny a v určitý čas rychlost tečná. Každým bodem v objemu tekutiny prochází vždy jen jedna proudnice. Pokud je v bodě nulová rychlost nachází se zde kritický bod. Při ustáleném neboli stacionárním proudění splyne proudnice a trajektorie v jednu křivku. [8]

Dalšími pojmy, které kinematika tekutin zavádí, jsou proudová trubice, proudové vlákno a emisní čára. Proudová trubice vznikne tak, že do objemu vložíme uzavřenou křivku tak, že jí všechny proudnice protínají jen jednou a s žádnou proudnicí nesplývá. Tekutina uvnitř proudové trubice teče jakoby byla hranice pevná a nepropustná, jelikož rychlost proudící tekutiny je na tuto plochu vždy tečná. Proudové vlákno je pak obsahem proudové trubice. Při stacionárním proudění je tvar proudové trubice stálý a tudíž objem tekutiny uvnitř této trubice je také stálý. Při nestacionárním proudění však dochází jak ke změně tvaru proudové trubice, tak i ke změně objemu touto trubicí ohraničeného. Posledním pojmem pak jsou emisní čáry. Ty spojují všechny částice, které prošly daným místem v tekutině a slouží pro zobrazení pohybu tekutin.

Pro pohyb tekutiny jsou v kinematice tekutin zavedeny důležité věty. Tyto věty jsou **1. věta Helmholtzova**, **2. věta Helmholtzova** a **Kelvin Thomsonova věta. První věta Helmholtzova** říká, že každý pohyb tekutiny v okolí každého bodu lze rozložit na tři elementární pohyby. Těmito pohyby jsou translace, rotace a deformace. Výsledný vektor rychlosti je pak dán třemi členy, kde první člen zastupuje translaci, druhý rotaci a třetí je pak deformace. Výsledný zápis je pak popsán podle [8] rovnicí

$$\vec{u}(\vec{x}',t) = u_i(\vec{x},t) + \vec{\omega} \times (\vec{x}' - \vec{x}) + \dot{e}(\vec{x}' - \vec{x}) . \qquad (2.2)$$

$$\vec{\omega} = \begin{pmatrix} \dot{\omega}_{32} \\ \dot{\omega}_{13} \\ \dot{\omega}_{21} \end{pmatrix} = \frac{1}{2} \nabla \times \vec{u} \quad ,$$

kde u je rychlost, x a x' jsou polohové vektory dvou blízkých částic v tekutině, $\vec{\omega}$ je vektor úhlové rychlosti rotace, $\dot{e}$ je pak tenzor rychlosti deformace 2. řádu.[8]

Další větou je **2. věta Helmholtzova,** ta se pak zabývá vířivostí a říká, že v každém průřezu vírové trubice a v daném okamžiku je intenzita víru stejná. Z toho vyplývá, že vírová trubice musí být uzavřená uvnitř objemu nebo procházet hranicí. Vírová trubice je definována analogicky jako proudová trubice, jedná se tedy o uzavřenou křivku. Dalším pojmem, který je třeba ještě vysvětlit je **intenzita víru** to vyžaduje zavedení **vířivosti Ω**. Vír rychlosti je odvozen z vektoru $\vec{\omega}$ dle následujícího vztahu $\vec{\Omega} = 2\vec{\omega} = \nabla \times \vec{u}$. Pokud je Ω=0 v celém objemu tekutiny, pak se jedná o nevířivé proudění, v opačném případě je proudění vířivé. Nyní se přesuneme zpět k intenzitě víru, ta je dána dle [8] rovnicí

$$\mu = \int_S \vec{\Omega}\, \vec{n}\, dS \quad . \tag{2.3}$$

Po odvození intenzity víru na vstupu a výstupu trubice by nám vyšlo, že intenzity se musí na vstupu a výstupu rovnat.

Poslední větou je pak **Kelvin Thomsonova věta.** Ta říká, že časová derivace cirkulace rychlosti podél uzavřené křivky se rovná cirkulaci zrychlení podél téže křivky. Cirkulace rychlosti je definována rovnicí

$$\Gamma = \oint_l \vec{u} \cdot \vec{dl} = \int_S \nabla \times u \cdot \vec{n}\, dS \tag{2.4}$$

a výsledek Kelvin Thomsonovy věty rovnicí

$$\frac{d\Gamma}{dt} = \oint a_i\, dx_i \,, \tag{2.5}$$

kde a je cirkulace zrychlení.

2.3 Dynamika ideálních tekutin

CFD simulace se zabývají numerickým řešením proudění tekutiny, proto v mé práci bude věnována velká pozornost mechanice tekutin a v následujících kapitolách i numerickému řešení. Ideální tekutiny jsou tekutiny, v nichž nepůsobí vnitřní tření. Dynamika tekutin se zabývá prouděním tekutiny při působení vnějších sil, pohyb dokonalé tekutiny je popsán pěti veličinami. Mezi tyto veličiny patří tři složky rychlosti

částic tekutiny a dvě termodynamické veličiny, kterými jsou tlak a hustota. Další veličiny se pak dopočítají ze stavové rovnice. Dynamika tekutiny se řídí třemi základními formulacemi, které jsou definovány jako zákony zachování:

- **zákon zachování hmoty (rovnice kontinuity)**
- **zákon zachování hybnosti (Eulerovy pohybové rovnice)**
- **zákon zachování energie (Bernoulliho rovnice)**

Z každého zákona zachováni se odvodí základní rovnice popisující proudění tekutiny. Ze zákona zachování hmoty se odvodí rovnice kontinuity, ze zákona zachováni hybnosti pak Eulerovy pohybové rovnice a nakonec ze zákona zachování energie se odvodí Bernoulliho rovnice.[4]

Prvním zákonem je zákon zachování hmoty a z něj odvozená rovnice kontinuity. Pro kontrolní objem musí být hmotnost tekutiny konstantní, z toho vyplývá, že její celková změna bude nulová. V kontrolním objemu mohou nastat dvě změny hmotnosti. Jednou ze změn hmotnosti je lokální změna v kontrolním objemu vlivem stlačitelnosti tekutiny. Druhá změna je pak konvektivní změna hmotnosti, která vzniká rozdílným přítokem a odtokem hmotnosti z kontrolního objemu. Obě tyto změny musí dávat nulovou změnu hmotnosti. To nastane, když jsou obě dílčí změny stejně velké, ale s rozdílným znaménkem. Rovnici kontinuity je možné definovat tak, že rozdíl vstupující hmotnosti a vystupující hmotnosti z kontrolního objemu je roven hmotnosti, která se v tomto kontrolním objemu akumuluje.[8]

Při odvození rovnice postupujeme dle [8] tak, že zvolíme kontrolní objem V, vhodně zvoleným objemem může být hranolek se stranami dx, dy a dz. Hranici tohoto objemu označíme jako dS a nazveme jí kontrolní plochou, přes kontrolní plochou protéká tekutina. Hmotnost tekutiny m, která proteče přes tuto plochu definujeme rovnicí

$$m = \int_S \rho \vec{u} \vec{n} dS = \int_S \rho u_i n_i dS \,. \tag{2.6}$$

Pokud je $\vec{n}$ normála, lze rozlišit vtok a výtok znaménkem. Pro vtok si zavedeme „+“ a pro odtok „-“.

Dále se zavede úbytek hmotnosti tekutiny v objemu V, který je definován vztahem

$$-\frac{\partial}{\partial t}\int_V \rho dV \,. \tag{2.7}$$

Pokud nejsou nikde v celém objemu ani zřídla ani propady hmoty, musí být součet obou předchozích vztahů (2.6) (2.7) roven 0, navíc rovnici (2.6) lze pomocí Gaussovy věty převést z plošného integrálu na objemový, čím získáme rovnici

$$-\int_V \frac{\partial \rho}{\partial t} dV = \int_V \nabla \cdot (\rho \vec{u}) dV \,. \tag{2.8}$$

Zároveň na obou stranách rovnice dostaneme objemové integrály, což nám pomůže v dalším postupu. Toho využijeme tak, že oba členy dáme pod jeden integrál a položíme ho roven nule rovnice

$$\int_V \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) \right] dV = 0 \,. \tag{2.9}$$

Jelikož se má integrál rovnat nule, musí se rovnat nule i integrand

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0 \,. \tag{2.10}$$

Pro nestlačitelné tekutiny je ρ konstantní, tím pádem se v rovnici kontinuity člen s časovou derivací rovná nule a výsledná rovnice je pak zapsána rovnicí

$$\nabla \cdot \vec{u} = 0 \,. \tag{2.11}$$

Zároveň se tento tvar označuje jako rovnice nestlačitelnosti.

Ideální tekutina se začne pohybovat tehdy, pokud dojde k narušení rovnováhy mezi vnějšími objemovými silami a vnitřními silami (tlakovými silami uvnitř tekutiny). V klidovém stavu je součet vnitřních a vnějších sil nulový a je dán rovnicí rovnováhy

$$-\frac{1}{\rho} \nabla p + \vec{G} = 0 \,. \tag{2.12}$$

První člen prezentuje hustotu vnitřních sil, druhý pak hustotu vnějších objemových sil.

Pro odvození Eulerových rovnic dle [8] si opět zavedeme kontrolní objem ve formě hranolku o stranách dx, dy, dz. Na kontrolní objem působí stejně jako v hydrostatice vnitřní tlaková síla dF_p a vnější tlaková síla dF_m. Podle Newtonova zákona výslednice rozdílu těchto sil udává proudící tekutině zrychlení $\vec{a}$. Původní rovnice rovnováhy se pak změní na tvar

$$-\frac{1}{\rho} \nabla p + \vec{G} = \vec{a} = \frac{d\vec{u}}{dt}, \tag{2.13}$$

tento tvar lze přeformulováním derivace rychlosti zapsat rovnici ve tvaru

$$\frac{d\vec{u}}{dt} = \frac{\partial \vec{u}}{\partial t} + \sum_i \frac{\partial u_i}{\partial x_i} * u_i$$

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} + \frac{1}{\rho}\nabla p - \vec{G} = 0 . \qquad (2.14)$$

V Eulerových rovnicích hydrodynamiky je celkem pět neznámých, tři složky rychlosti, hustota ρ a tlak p. Eulerova rovnice hydrodynamiky je nelineární parciální diferenciální rovnice, její integrace je obtížná i časově náročná, v současné době se řeší numericky.

Při proudění dokonalé tekutiny působí na její částice síly, které při posunutí po dráze konají práci. Pro odvození Bernoulliho rovnice vycházíme ze zákona zachování energie. Vychází se z Eulerových pohybových rovnic, ve kterých se upraví člen obsahující derivace rychlosti podle polohy a zavedou se následující předpoklady. Těmito předpoklady jsou, že tekutina je barotropní (hustota je funkcí tlaku), vnější objemové síly jsou potenciální a zavede se potenciál *U*. Nakonec upravíme Eulerovy rovnice tak, aby šly integrovat podél proudnice. Toho docílíme tak, že Eulerovy rovnice s upravenými časovými derivacemi vynásobíme vektorem elementu oblouku proudnice $d\vec{x}$. Na konci těchto všech operací získáme rovnici

$$\frac{\partial u_i}{\partial t} dx_i + \frac{\partial}{\partial x_i}\left(\frac{u^2}{2}\right) dx_i + \frac{1}{\rho}\frac{\partial p}{\partial x_i} dx_i + \frac{\partial U}{\partial x_i} dx_i = 0 . \qquad (2.15)$$

Integrací této rovnice získáme rovnici

$$\int \frac{\partial u_i}{\partial t} dx_i + \frac{u^2}{2} + U + \int \frac{dp}{\rho} = konst . \qquad (2.16)$$

Pokud rovnici (2.16) použijeme pro stacionární proudění nestlačitelné tekutiny je časová derivace rychlosti rovna 0, potenciál *U* se v gravitačním poli zapíše $U = gz$, člen pravé strany se upraví $\int \frac{dp}{\rho} = \frac{p}{\rho}$ a celou rovnici vynásobíme hustotou ρ. Získáme známý tvar Bernoullioho rovnice

$$\frac{1}{2}\rho u^2 + g\rho z + p = konst . \qquad (2.17)$$

Jednotlivé členy rovnice pak prezentují kinetickou, polohovou a tlakovou energii proudící kapaliny. Jejich součet je pak roven celkové mechanické energii kapaliny, která podle Bernoulliho rovnice je v každém průřezu jedné zvolené trubice konstantní. Bernoulliho rovnice (2.17) platí pro proudovou trubici, ve které je rychlost rovnoměrně rozložena. Při nerovnoměrném rozložení rychlosti se musí zvolit proudová trubice o dostatečně malém průřezu, aby rozdíl rychlostí v daném průřezu proudové trubice byl velmi malý.[8]

2.4 Navier-Stokesovy rovnice

Na úvod této kapitoly si připomeneme, jak se od sebe odlišuje ideální a viskózní kapalina. Jak již bylo zmíněno, hlavním rozdílem je ten, že v ideální tekutině neexistuje vnitřní tření, tečné napětí je tedy nulové. Naproti tomu viskózní tekutina má určitou viskozitu a tečné složky tenzoru napětí jsou nenulové. Výsledný tenzor napětí obsahuje jak diagonální (normálové napětí), tak mimodiagonální složky (tečné napětí) a má tvar matice

$$\tau(\vec{x},t)=\begin{pmatrix}\tau_{11} & \tau_{12} & \tau_{13}\\ \tau_{21} & \tau_{22} & \tau_{23}\\ \tau_{31} & \tau_{32} & \tau_{33}\end{pmatrix}. \tag{2.18}$$

Pro izotropní tekutinu je navíc tenzor napětí symetrický ($\tau_{ij}=\tau_{ji} \quad \forall i,j$), většina tekutin se považuje za izotropní.

Nyní se zaměříme na to, jak se změna tenzoru napětí projeví v pohybových rovnicích a odvodíme pohybové rovnice, které se pro Newtonovskou viskózní tekutinu nazývají **Navier-Stokesovy** rovnice. Při odvozeni dle [8] se vychází z rovnice rovnováhy

$$\frac{\partial \tau_{ij}}{\partial x_j}+F_i=0. \tag{2.19}$$

Rovnice rovnováhy se rozšíří o člen vyjadřující změnu rychlostního pole, který je způsoben nerovnováhou sil. Po úpravách vzniknou obecné pohybové rovnice tekutiny

$$\frac{\partial u_i}{\partial t}+u_j\frac{\partial u_i}{\partial x_j}=G_i+\frac{1}{\rho}\frac{\partial \tau_{ij}}{\partial x_j}\ , \tag{2.20}$$

kde G_i jsou vnější síly.

Aby z obecných pohybových rovnic vznikly **Navier-Stokesovy** rovnice musí tenzor τ splňovat **Stokesovy postuláty**. Stokes definoval pro newtonovskou kapalinu pět postulátů týkajících se tenzoru napětí. Těmito postuláty jsou:

1. $\tau=-pE+\tau'$, kde E je jednotková matice
2. τ' je spojitá tenzorová funkce rychlosti deformace, navíc nesmí být závislá na jiných kinematických veličinách a není explicitně závislá na poloze a času
3. Tekutina je izotropní, tekutina má stejné vlastnosti ve všech směrech
4. Pokud je tenzor rychlosti deformace nulový $\dot{e}=0$, pak na tekutinu působí jen tlakové síly
5. Vztah mezi τ' a $\dot{e}$ je lineární

Po splnění **Stokesových postulátů** se tenzor napětí změní na tvar

$$\tau = -pE + \lambda\nabla\cdot uE + 2\mu\dot{e} = -\left(-p + \lambda\nabla\cdot\vec{u}\right)E + 2\mu\dot{e}\ , \qquad (2.21)$$

kde E je jednotková matice a μ,λ jsou dynamická a druhá viskozita.

Dosazením upraveného tenzoru napětí (2.21) do obecné pohybové rovnice (2.20) získáme pro Newtonovskou tekutinu **Navier-Stokesovy rovnice**

$$\rho\frac{\partial\vec{u}}{\partial t} + \rho\left(\vec{u}\cdot\nabla\right)\vec{u} = \rho\vec{G} - \nabla p + \nabla\left(\lambda\nabla\cdot\vec{u}\right) + \nabla\cdot\left(2\mu\dot{e}\right). \qquad (2.22)$$

Pro nestlačitelnou kapalinu se člen $\nabla\cdot(u) = 0$ a Navier-Stokesovy rovnice se zjednoduší na tvar

$$\frac{\partial\vec{u}}{\partial t} + \vec{u}\left(\nabla\vec{u}\right) = \vec{G} - \frac{1}{\rho}\nabla p + \nu\left(\Delta u\right), \qquad (2.23)$$

kde ν je kinematická viskozita.

Navier-Stokesovy rovnice jsou v dnešní době nejčastěji používány při numerických výpočtech proudění tekutin. Jedná se o parciální diferenciální rovnice odvozené již na počátku 19. století. Nesmíme však zapomenout, že v obecném případě není dosud prokázána existence a jednoznačnost řešení úlohy proudění pro určité zadané počáteční podmínky. Tuto problematiku se zatím nepodařilo vyřešit a je to jeden ze sedmi otevřených matematických problémů tohoto tisíciletí. Nejvíce se to pak projevuje u turbulentního proudění, které samo osobě představuje velmi těžkou disciplínu, kde nejsou ještě objasněny všechny principy a chování tohoto proudění.

2.5 Laminární a turbulentní proudění

Po odvození pohybových rovnic už známe, podle jakých zákonitostí tekutina proudí. Nyní se tedy zaměříme na odlišení jednotlivých druhů proudění. Touto problematikou se již na konci 19. století zabýval anglický fyzik Osbone Reynolds. Sestrojil jednoduchý přípravek, který se skládal z duté průhledné trubice kruhového průřezu, z nádobky s barvivem a nádrže (viz obr. 2.3).

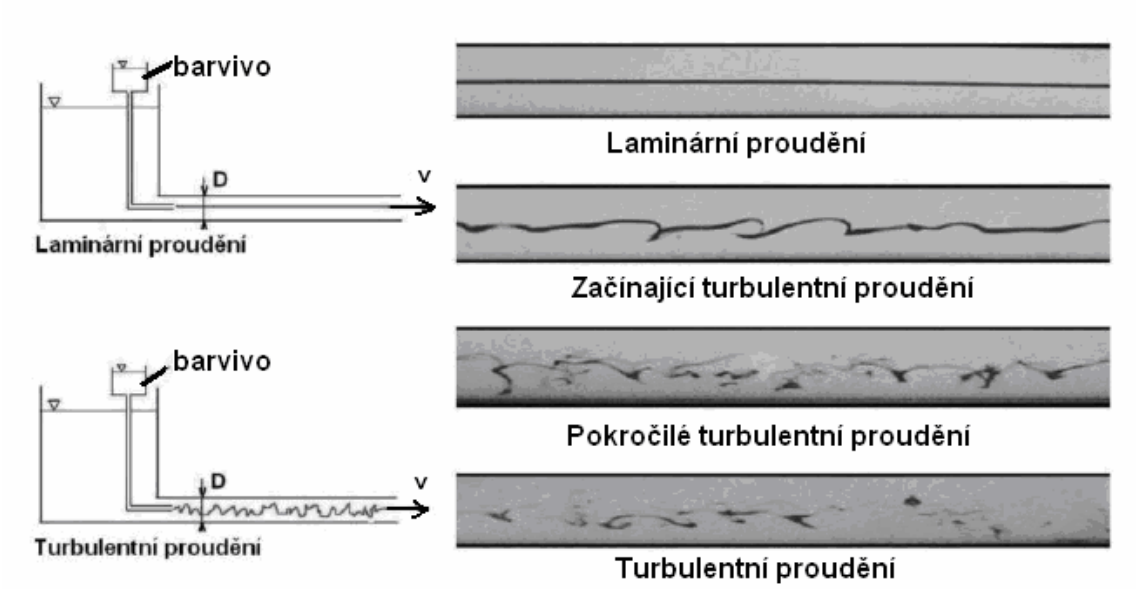
Na tomto přípravku pak reguloval rychlost průtoku kapaliny, do které se na vstupu do trubice dávkovalo barvivo. Při nízkých rychlostech se barvivo nerozmývalo a vytvořilo v proudící kapalině rovnou čáru. Takovéto proudění se nazývá laminární.

Při zvýšení rychlosti proudící tekutiny se barvivo v určitý okamžik začalo rozmývat v kapalině a postupně se barvivo rozmísilo do celého průřezu trubice. Při svém pokusu si také Reynolds všiml, že k přechodu mezi laminárním a turbulentním

prouděním dochází, až po překročení určité rychlosti. Vytvořil tedy novou bezrozměrnou proměnou, která se dnes nazývá Reynoldsovo číslo a značí se *Re*. Toto číslo je dáno vztahem

$$\mathrm{Re} = \frac{u \cdot D}{\nu}, \tag{2.24}$$

který dává součin rychlosti proudění a průměru trubice do poměru s kinematickou viskozitou. Dále také zavedl kritické Reynoldsovo číslo, které udává, kdy laminární proudění přejde v turbulentní proudění. Hodnota kritického Reynoldsova čísla pro proudění v uzavřeném kanálu kruhového průřezu (trubce) je přibližně 2300. [6]

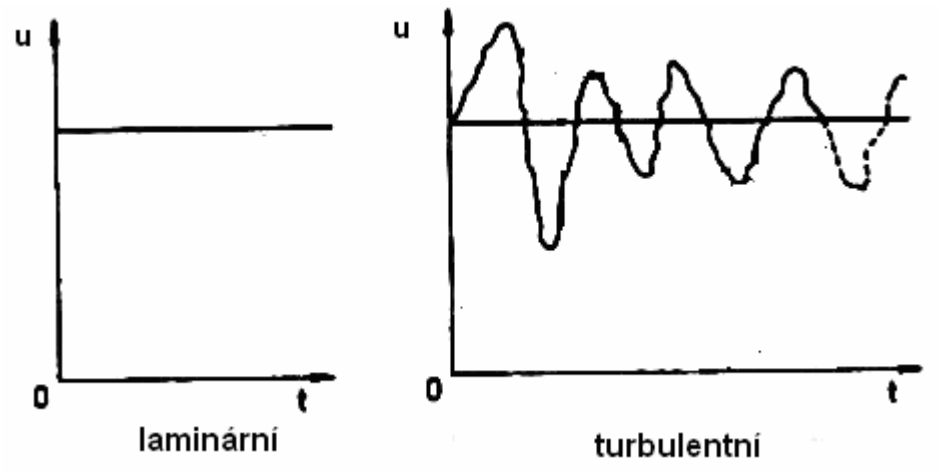


Obr. 2.3: Reynoldsuv pokus [4]

Reynolds také zjistil, že přechod mezi laminárním a turbulentním prouděním není skokový, ale je dán intervalem Reynoldsových čísel. Reynolds svým zkoumáním proudění tekutiny a objevením turbulentního proudění položil základy pro výzkum právě tohoto proudění. Výzkum turbulence není ukončen ani v této době.

Přechod laminárního proudění v turbulentní závisí kromě Reynoldsova čísla i na dalších okolnostech, jako je drsnost stěn a stupeň turbulence přitékajícího proudu. Podle současných pozorování je jednou z příčin přechodu z laminárního proudění na proudění turbulentní nestabilita laminárního proudění. Experimentálně bylo ověřeno, že laminárního proudění lze dosáhnout v potrubí i při velmi vysokých Reynoldsových číslech, řádově 10^4.

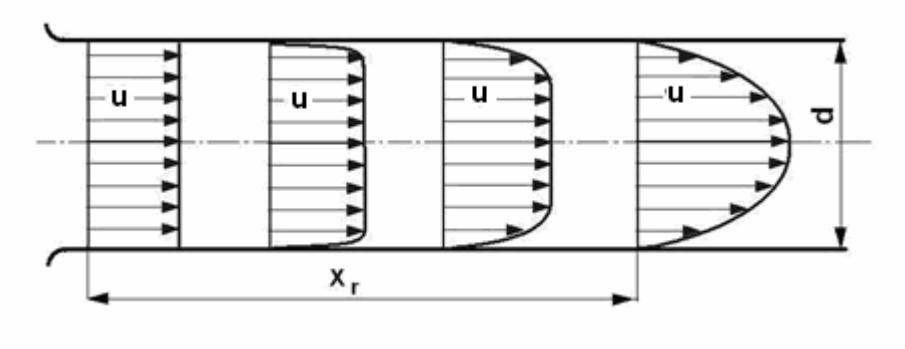
Hlavní rozdíl mezi laminárním a turbulentním prouděním je zřejmý na časovém průběhu rychlosti proudící tekutiny (viz obr. 2.4). U laminárního proudění nedochází k promíchávání vrstev tekutiny a střední rychlost je konstanta. U turbulentního proudění, které má časově proměnou hodnotu rychlosti, se vyskytuje turbulentní složka rychlosti, která je malá a má časově proměnou velikost a také směr [8], [9].



Obr. 2.4: Časový průběh rychlosti [4]

Laminární proudění se oproti turbulentnímu mnohem jednodušeji modeluje. Pro modelovaní laminárního proudění se používají Navier-Stokesovy rovnice bez jakýchkoliv úprav. V některých jednoduchých případech lze řešení získat dokonce analyticky. Takovým příkladem je třeba proudění v potrubí o konstantním průřezu. S laminárním prouděním se nejčastěji setkáme při proudění kapaliny v potrubích o malých průměrech, při malých rychlostech proudící tekutiny a u proudění tekutin s vysokou viskozitou.

Laminární proudění se dá popsat tak, že jednotlivé vrstvy tekutiny po sobě navzájem kloužou a nepromíchávají se. Při řešení laminárního proudění se uplatňuje Newtonův vztah, který pro newtonovské kapaliny udává lineární závislost smykového napětí na gradientu rychlosti. Koeficient lineární závislosti je pak dán přímo viskozitou tekutiny. A jelikož většina tekutin, se kterými se v životě setkáme je newtonovských, odpovídá teoretické řešení výsledkům naměřeným metodami experimentální mechaniky tekutin. Při laminárním proudění se v potrubí za přechodovou zónou ustanoví rychlostní profil (viz obr. 2.5) [6].



Obr. 2.5: Rychlostní profil při laminárním proudění [4]

Většina proudění, které pozorujeme kolem nás, je turbulentní. Při tomto proudění dochází k náhodným změnám veličin jako je rychlost, tlak, teplota, hustota a další. Turbulentní proudění si můžeme představit jako náhodný pohyb částic tekutiny, tento pohyb částic se skládá z uspořádaného pohybu a z náhodných fluktuací. Nesmíme zapomenout, že turbulentní proudění je pohyb kontinua. U turbulentního proudění vzniká tečné napětí jako u laminárního proudění, ale není určeno pouze viskozitou a gradientem rychlosti, ale mimo jiné i změnou hybnosti makroskopických částeček při promíchávání sousedních vrstev. Tento neuspořádaný pohyb vyvolá tzv. přídavná turbulentní napětí, také nazývaná Reynoldsova napětí. Z těchto vlastností je patrné, že v porovnání s laminárním prouděním je turbulentní proudění složitější a jeho matematický model mnohem komplikovanější. [4]

Při turbulentním proudění lze pro tečné napětí použít Boussinesqovu hypotézu. Tato hypotéza předpokládá, že stejně jako u laminárního proudění platí pro smykové napětí Newtonův vztah (2.1). Turbulentní napětí τ_t jsou úměrné gradientu střední rychlosti a turbulentní viskozitě η_t. Pro obecné 3D proudění je pak turbulentní napětí dáno vztahem

$$\tau_t = -\rho\overline{u_i' u_j'} = \eta_t\left(\frac{\overline{\partial u_i}}{\partial x_j} + \frac{\overline{\partial u_j}}{\partial x_j}\right) - \frac{2}{3}k\rho\delta_{ij} \quad , \qquad (2.25)$$

$$k = \frac{1}{2}\overline{u_j' \cdot u_j'} \qquad \textit{kinetická} \text{ energie } \textit{turbulence}$$

kde hodnoty s pruhem jsou středované, bližší informace, jak se získají středované veličiny jsou v následující kapitole 3.1.

Turbulentní viskozita η_t není konstantou jako u laminárního proudění, ale je funkcí závislou na stavu proudící tekutiny, poloze v tekutině a tvaru rychlostního pole.

V obecném případě je výsledné tečné napětí dáno součtem napětí laminárního a turbulentního proudění. [9]

Nyní se zaměříme na samotné turbulence, podle kterých se dané proudění nazývá. Obecná definice samotné turbulence nebyla dosud přijata, obvykle se nahrazuje výčtem vlastností, podle kterého pak můžeme identifikovat turbulentní proudění.

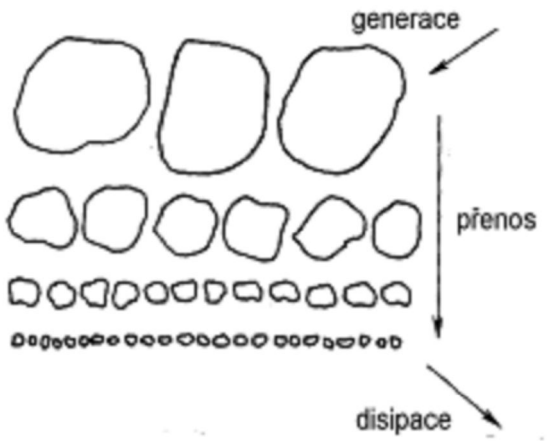
Prvním z těchto parametrů je náhodnost. **Náhodnost** nám říká, že turbulentní proudění je nepředvídatelné a i malé počáteční poruchy proudění mohou být zesíleny tak, že již není možno předpovědět další vývoj těchto poruch. To způsobí, že není možné předpovědět přesně chování konkrétního turbulentního proudění. Ale lze ho předpovědět ve statistickém smyslu.

Další parametr je pak **difuzivita**. V turbulentním proudění dochází k rychlejšímu promíchávání transportovaných skalárních veličin (koncentrace a teplota), než při molekulární difusi. Turbulence je charakterizována zvýšeným promícháváním. Intenzita tohoto promíchávání může být až o několik řádů větší než je tomu při molekulární difusi.

Nyní se zaměříme na **vířivost.** Turbulentní proudění je charakterizováno vysokými hodnotami rotační složky rychlosti neboli vířivostí. Pole vířivosti je nehomogenní a mění se s časem. Víry, které vznikají spontánně, jsou charakterizovány širokou škálou délkových měřítek. Maximální velikost víru je omezena velikostí oblasti, kudy tekutina proudí, a minimální velikost víru je pak omezena takovou velikostí víru, kdy dochází k **disipaci**, která je ovlivněna viskozitou. Disipace pak znamená, že kinetická energie pohybující se tekutiny ve víru je v malých vírech při jejich zániku vlivem viskózního tření přeměněna na teplo. Pro zachování turbulentního proudění se neustále odebírá energie z hlavního proudu tekutiny. To se děje v oblasti velkých měřítek (velkých vírů), malé struktury (malé víry) už nejsou schopny odebírat energii z proudící tekutiny. Energie je pak dále předávána pomocí kaskádového přenosu energie k menším měřítkům (menším vírům) až do té doby, než dojde k disipaci.

Nyní si ve stručnosti popíšeme kaskádový přenos energie. Při rozpadu velkých vírů se uplatňuje zákon zachování energie. Vír má moment setrvačnosti J a otáčí se úhlovou rychlostí ω. Menší víry po rozpadu velkého víru mají menší hmotnost a tedy i moment setrvačnosti než vír původní, a proto musí vzrůst úhlová frekvence. V turbulentním proudění pozorujeme, že velké víry mají frekvenci otáčení malou, naopak víry malé se otáčejí mnohem rychleji. Výše popsaný proces však dále pokračuje,

protože vzniklé malé víry se v důsledku stejného mechanismu dále rozpadají (viz obr. 2.6). [9],[4]



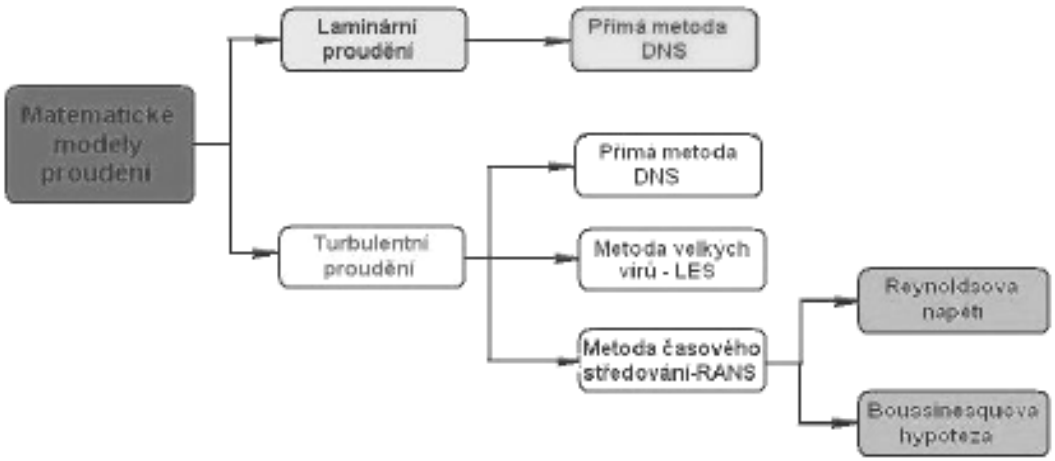
Obr. 2.6: Kaskádní přenos energie u turbulentního proudění [4]

Posledním parametrem je **nelinearita.** Vývoj i interakce jednotlivých vírů v turbulentním poli lze popsat pouze nelineárním matematickým modelem.

3 CFD

3.1 Matematické modely proudění tekutiny

CFD (*Computational fluid dynamics*) je vědní disciplínou, která se zabývá numerickým řešením proudění tekutiny. CFD volně v překladu znamená výpočetní mechanika tekutin. Toto odvětví rozvíjelo společně s rozvojem výpočetní techniky. Zpočátku nebyl výpočetní výkon a velikost paměti v počítačích dostatečný, aby se dalo modelovat proudění viskózních tekutin. Postupem času zvyšující se výpočetní výkon počítačů umožnil modelování nejenom proudění laminárního, ale také modelování turbulentního proudění. Při výpočtu proudění tekutiny, kdy se tekutina bere jako kontinuum se pro vytvoření matematického modelu vychází převážně z Navier-Stokesových rovnic a rovnice kontinuity, v některých případech proudění se uplatňují i další rovnice, odvozené ze zákonů zachování. Pro řešení se využívá hned několik druhů matematický modelů (viz obr 3.1).



Obr. 3.1: Druhy matematický modelů pro řešení proudění [4]

Pro laminární proudění se nejčastěji využívá metoda přímé simulace DNS (Direct Numerical Simulations), tedy přímé diskretizaci N-S rovnic. Nevýhoda této metody je, že numerické řešení vyžaduje pro střední a vysoká Re velice jemnou sít, jelikož časová a prostorová diskretizace musí být schopna zachytit celé spektrum vírových struktur. Jemnost sítě je závislá na Reynoldsově čísle, závislost počtu uzlů sítě na Reynoldsově číslu je udávána v rozmezí od $Re^{9/4}$ do Re^{3} pro prostorově složitou geometrii, a vysoká Reynoldsova čísla je tato závislost dokonce Re^{6}. Z této závislosti je patrné, že se přístup uplatní pro proudění s malými Reynoldsovými čísly. Další důsledek, který tato závislost

vyvolá je ten, že s rostoucím počtem uzlů se zmenšuje prostorový diskretizační krok. To má za následek zkracovaní i časového diskretizačního kroku díky CFL (Courant–Friedrichs–Lewy) podmínce. CFL je nezbytnou podmínkou pro konvergenci numerického řešení parciálních diferenciálních rovnic při použití explicitního diskretizačního schématu. Podmínka je prezentována někdy jako Courantovo číslo a je dána předpisem

$$\frac{u \cdot \Delta t}{\Delta x} = Co < 1$$
$$\frac{u_x \cdot \Delta t}{\Delta x} + \frac{u_y \cdot \Delta t}{\Delta y} + \frac{u_z \cdot \Delta t}{\Delta z} = Co < 1 \quad , \qquad (3.1)$$

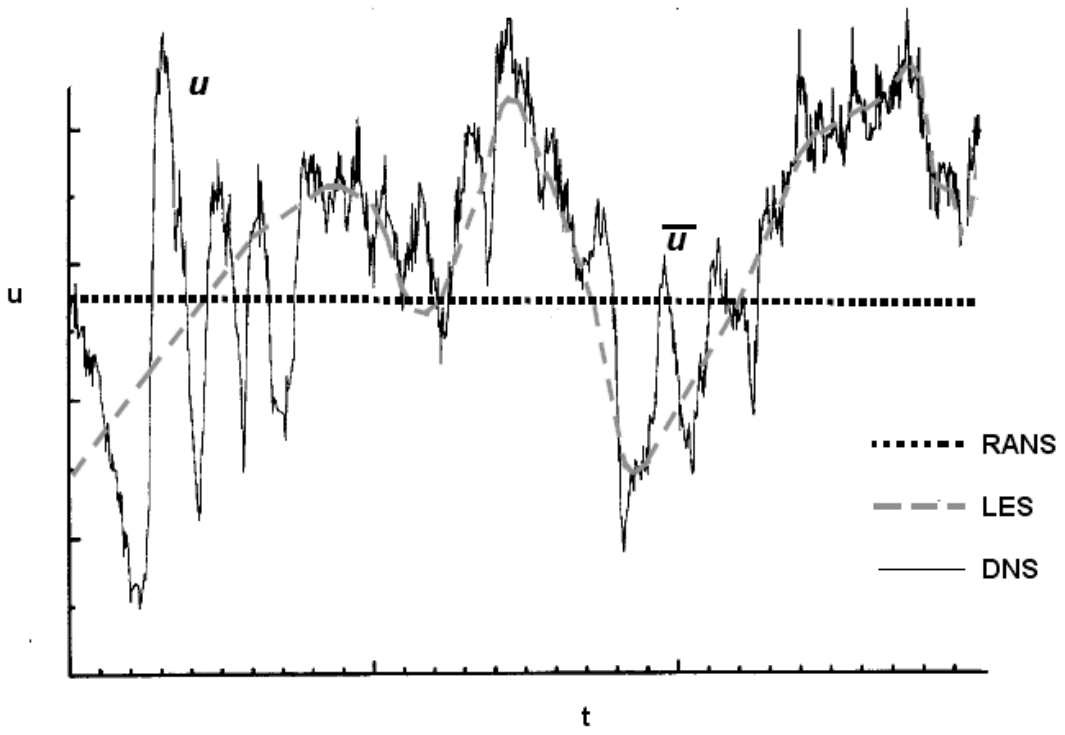
kde Δt je krok časové diskretizace, Δx je krok prostorové diskretizace a *u* je rychlost proudění. Aby došlo vždy ke konvergenci numerického řešení, volí se nejčastěji Co v rozsahu 0.5-1.

Další metodou, která se používá pro matematické modelování turbulence, je metoda LES (Large Eddy Simulation), v českém překladu simulace velkých vírů. Tato metoda je založena na tom, že se řešený problém rozdělí na dvě části. Jedna část zahrnuje velké víry až po určitou mez a druhá část pak obsahuje malé víry. Mez víru je pak určena velikostí diskretizačního kroku sítě. Obě části jsou však spolu provázány a nelze je tedy řešit nezávisle na sobě. Nejčastěji se používá pro řešení velkých vírů přímá metoda DNS a na řešení malých se pak používá tzv. *subgrid model*. Zatím se metoda LES moc často nepoužívá, přestože nabízí kvalitní výsledky v oblasti akademického použití a není tak omezená výpočetním výkonem počítačů jako metoda DNS .

Pro řešení většiny inženýrských aplikací se využívají metody odvozené z RANS (Reynolds Averaged Navier-Stokes Equations) neboli v českém překladu Reynoldsovo středování Navier-Stokesových rovnic. Jedná se o statistickou metodu, která je založena na časovém středování fyzikálních veličin a středování rovnic. Středování rovnic docílíme dosazením středovaných veličin do rovnice kontinuity a Navier-Stokesových rovnic. Středováne rovnice kontinuity a Navier-Stokesovy rovnice se pak nazývají Reynoldsovy rovnice. Středování fyzikálních veličin (rychlosti, tlaku, teploty atd.) se pak provádí tak, že se okamžitá hodnota veličin rozdělí na dvě složky dle předpisu

$$u(x,t) = \overline{u}(x,t) + u'(x,t) \quad , \qquad (3.2)$$

kde $\overline{u}(x,t)$ je střední hodnota veličiny a $u'(x,t)$ složka obsahující náhodné fluktuace. Po středování je fluktuační složka nulová a zůstane pouze střední hodnota. Porovnání výsledků jednotlivých metod matematického modelování proudění je na obr. 3.2 [7].

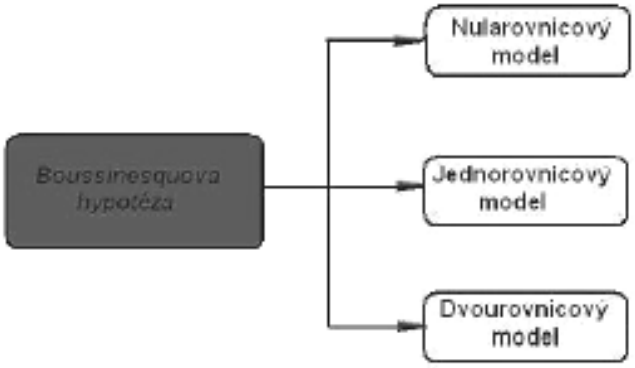


Obr. 3.2: Porovnání výsledku jednotlivých metod modelování turbulentního proudění [4]

Metod matematického modelování založených na RANS je několik, dále se dělí na dvě základní podskupiny. Hlavním rozdílem obou skupin je, jak se docílí toho, aby byl systém Reynoldsových rovnic uzavřen. U Reynoldsových rovnic je více proměnných než kolik se dá z daných rovnic vypočítat. Těmito skupinami tedy jsou metody založené na Boussinesquově hypotéze a metoda Reynoldsových napětí.

Na Boussinesquově hypotéze je založeno několik metod matematického modelování turbulentního proudění. Tuto hypotézu zavedl v roce 1877 Boussinesq a je obdobou Newtonova zákon vazkosti. Výsledná vazkost je pak dána součtem turbulentní vazkosti η_t a vazkosti tekutiny (molekulová vazkost). Turbulentní viskosita je však funkcí a nikoliv konstantou a je závislá na prostoru a čase. Pro zjednodušení se časová závislost vyruší. Dalším zjednodušením je, že výsledná celková vazkost je rovna pouze turbulentní vazkosti, molekulová vazkost je v porovnání s turbulentní zanedbatelná. Samotná turbulentní vazkost je však neznámou veličinou a neřeší tak uzavření Reynoldsových rovnic, proto se musí zavést další rovnice, které potřebuje právě k určení turbulentní viskosity. Z tohoto důvodu se používá další dělení metod

odvozených z Boussinesquovy hypotézy právě podle počtu přidaných rovnic(viz obr. 3.2). [4], [7]



Obr. 3.2: Matematické modely pro Boussinesquova hypotéza [4]

První případem je nularovnicový model nebo algebraický model**.** Tento model je založen na zavedení směšovací délky L_m, kterou zavedl Prandtl. Směšovací délka pak udává vzdálenost, kterou vír urazí v mezní vrstvě, než zanikne. Tato směšovací délka slouží k určení turbulentní viskozity dle vztahu

$$\eta_t = \rho L_m{}^2 \frac{\partial \overline{u}}{\partial x}. \tag{3.3}$$

Algebraické modely jsou určeny především pro dvourozměrné proudění v mezní vrstvě nebo v úplavu. Pro prostorové řešení je tento model nevhodný. Jak se tato metoda postupně rozvíjela, bylo odvozeno mnoho modifikovaných algebraických modelů. Některé se používají v dnešní době v letectví, kde se podle nich stále modeluje obtékaní leteckého profilu.[7]

Dalším případem jsou pak modely složitější, které pro určení turbulentní viskosity používají transportní rovnice. První skupinu tvoří model jednorovnicový model. Jednorovnicový model používá transportní rovnici pro určení turbulentní energie. Nejčastěji se definuje v kinematickém tvaru dle vztahu

$$k = \frac{1}{2}\frac{\overline{\rho u_i' u_j'}}{\overline{\rho}}. \tag{3.4}$$

Zavedení této kinetické turbulentní energie se projeví ve výpočtu turbulentní viskosity, která je pak definována následujícím vztahem

$$\eta_t = C_\nu \overline{\rho} \sqrt{k} L_\mu \ , \tag{3.5}$$

kde $\sqrt{k}$ je rychlostní měřítko turbulentního pohybu a L_μ je délkové měřítko.

Použití samostatného jednorovnicového modelu je v praxi velice omezené. Je vhodný pro výpočet tenkých smykových vrstev (mezní vrstva a proudění v blízkosti stěny). Nejčastěji se používá tzv. dvouvrstvý model, který rozděluje oblast na dvě části. První část se nachází v blízkosti stěn a uplatňuje se zde jednorovnicový model a v oblasti vzdálené od stěn se naopak využívá dvourovnicový model. Výhodou tohoto spojení je nižší počet uzlů sítě, než kdyby byl použit jen dvourovnicový model.

Pro výpočet běžných technických problémů se využívají dvourovnicové modely. Výhodou tohoto modelu oproti jednorovnicovému je jeho použitelnost i pro prostorové proudění. To je způsobeno tím, že zde není algebraický model pro délkové měřítko, které je závislé na vzdálenosti od stěny. Místo délkového měřítka je použita veličina vyjádřena pomocí transportní rovnice.[7]

Nejběžnějším a nejpoužívanějším dvourovnicovým modelem je model *k*-*ε*. V tomto modelu je nahrazeno délkové měřítko rychlostí disipace energie ε**,** která je dána transportní rovnicí. To má výhodu, že se v tomto modelu nevyskytuje algebraická závislost délkového měřítka. Naopak se délkové měřítko určí za pomoci rychlosti disipace energie. To se promítne i do určení turbulentní viskozity, která je dána vztahem

$$\eta_t = C_\mu \overline{\rho} \sqrt{k} L = C_\mu \overline{\rho} \frac{k^2}{\varepsilon} . \tag{3.6}$$

Nevýhoda tohoto modelu je, že je použitelný pouze v dostatečné vzdálenosti od stěny, kde proudění dosahuje velkých Re. U stěn totiž není turbulence izotropní vlivem tlumení fluktuačních rychlostí kolmých na stěnu. Z tohoto důvodu existuje několik možností úprav standardního *k*-*ε* modelu. Mezi tyto úpravy patří zavedení stěnových funkcí, úpravou na dvouvrstvý model nebo modifikace dvourovnicového modelu pro nízká turbulentní Reynoldsova čísla. Bližší podrobnosti k těmto modifikacím jsou k nalezení v [7].

Asi nejpoužívanější alternativou *k*-*ε* modelu je model *k*-*ω*. Tento model odstraňuje problémy s prouděním v blízkosti stěny tím, že místo rychlost disipace *ε* je nahrazena tzv. specifickou rychlostí disipace $\omega = \varepsilon / k$.

Poslední skupinou matematických modelů turbulence jsou metody založené na přímém modelování Reynoldsových napětí RSM (Reynolds Stress Models). Jsou výpočetně mnohem náročnější něž modely založené na Boussinesquově hypotéze. Tyto

modely používají rovnice pro výpočet Reynoldsových napětí odvozené přímo z Navier-Stokesových rovnic. Výhodou těchto modelů je poskytnutí dobrého řešení i pro komplexní proudění v geometricky složité oblasti.[7]

3.2 Numerické metody pro řešení proudění

V dnešní době se pro numerické řešení proudění nejčastěji používá metoda konečných objemů (Finite volume method). V některých výpočetních softwarech se lze setkat i s metodou konečných prvků (Finite element method), ale oproti metodě konečných objemů je zastoupení metody konečných prvků výrazně menší. A poslední metodou, která se používala hlavně v počátku rozvoje numerického řešení, byla metoda konečných diferencí. Výhodou této metody byla jednoduchost a snadná implementace, na druhou stranu, ale vyžaduje pravidelné pravoúhlé sítě, což je pro modelování proudění ve složité geometrii značně nevhodné a problematické. Hlavním důvodem, proč se proudění tekutiny řeší numericky, je to, že analytické řešení je složité a lze provést jen v akademických případech. Proto bylo nutné nalézt jiný způsob řešení problematiky proudění. S rozvojem výpočetní techniky se tímto způsobem stalo právě numerické řešení. Numerické řešení převádí rovnice popisující proudění na diskrétní tvar a tím umožňuje jejich řešení pomocí počítačů.

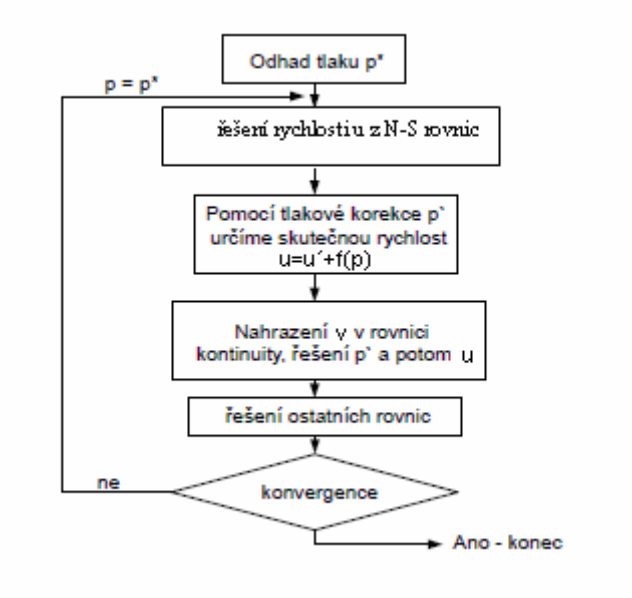
Všechny zmíněně metody jsou založeny na stejném principu a to, že se musí z řešení spojitého problému přejít na řešení problému diskrétního. To znamená přejít z popisu proudění pomocí parciálních diferenciálních rovnic na popis proudění soustavou lineárních rovnic. Toho se docílí vytvořením diskretizační sítě, která rozdělí oblasti na konečný počet elementů. To vytvoří v oblasti, ve které se řeší proudění, diskrétní body (uzly sítě). Pro všechny metody je také nutné předepsat okrajové podmínky na hranice .[17]

Metoda konečných objemů vychází z integrálního tvaru dané úlohy. Základem metody je, jak už samotný název napovídá, rozdělení řešené oblasti na systém vzájemně disjunktních kontrolních objemů. Tímto krokem je vyřešená prostorová diskretizace, ale také je nutné vyřešit časovou diskretizaci. Časová diskretizace lze provést dvěmi způsoby, a to za pomoci implicitního a explicitního schématu. Explicitní schéma je rychlejší a nevyžaduje tak velké nároky na paměť jako je tomu u implicitního schématu. Na druhou stranu je explicitní schéma méně stabilní a vyžaduje splnění CFL podmínky. Implicitní schéma naopak neklade žádné nároky na časovou diskretizaci a časový krok

může být mnohem delší, než je tomu u explicitního schématu. Největším omezením implicitního schématu je nutnost v každém časovém kroku řešit celou soustavu rovnic z důvodu provázanosti výpočtu hodnoty proměnné v uzlu pro nový časový krok.

Nyní přistoupíme k samotnému řešení počítaných veličin pomocí metody konečných objemů. Hodnoty počítaných veličin jsou definovány dvěma způsoby, a to buď ve středu konečného objemu (Cell-centred) nebo ve vrcholových uzlech (Cell-vertex). Je několik možných přístupů jak získat hodnoty počítaných veličin. První skupinu tvoří metody, které řeší soustavu lineárních rovnic získanou časovou a prostorovou diskretizací za pomoci schémat, které derivace nahradí diferencemi. Tyto metody se dělí na přímé, iterační (nepřímé) a multigrid, nevýhodou přímých metod je vysoká náročnost na pamět, jelikož se do paměti ukládá celá matice přesto, že je většina prvků nulových. Mezi přímé metody patří Gausova eliminace nebo LU rozklad. Naopak nepřímé metody jsou méně náročné na paměť, jelikož se do paměti ukládají pouze nenulové prvky. Mezi iterační metody patří Gauss-Seidelova metoda, Jacobiho metoda nebo metoda sdružených gradientů. Metoda Multigrid se od ostatních metod výrazně odlišuje, jelikož se pro výpočet používá více úrovňová síť, kde každá úroveň je oproti předešle hrubší nejčastěji se voli kh, kde k je úroveň h je velikost diskretizačního kroku sítě. [2]

Druhou skupinu pak tvoří metody, které jsou založeny na řešení N-S rovnic. Těmito metodami jsou metody sdružené nebo oddělené řešiče. Sdružené řešiče se vyznačují tím, že řeší N-S rovnici a zároveň tlak pomocí stavové rovnice. Rychlost konvergence je ovlivněna CFL podmínkou, sdružené řešiče jsou velice náročné na paměť. Druhou skupinu tvoří tzv. oddělené řešiče, které potřebují jak N-S rovnice tak rovnici kontinuity. Metody založené na tomto principu jsou SIMPLE, PISO a PIMPLE. První metodou je metoda SIMPLE (**S**emi-**I**mplicit **M**ethod for **P**ressure-**L**inked **E**quations). Metoda byla odvozená v roce 1972 a je založena na principu korekce tlaku a rychlosti obr. 3.3 [5]



Obr. 3.3: Diagram metody SIMPLE

Z důvodu nelinearity diferenciálních rovnic je nutné zavést relaxační faktor α, který určuje, kolik nové informace (hodnoty) přejde do řešení. Obecný zápis pro výpočet hodnot tlaku nebo rychlosti pak má předpis

$$\begin{aligned} p^{nový} &= p^{*} + \alpha_p p' \qquad 0 < \alpha_p > 1 \\ u^{nový} &= \alpha_u u^{k} + (1+\alpha_u) u^{k-1} \qquad 0 < \alpha_u > 1 \end{aligned} \quad , \qquad (3.7)$$

kde novou hodnotu rychlosti získáme kombinací hodnoty vypočtené z předešlého kroku a aktuálně vypočtenou hodnotou. Pro tlak se pak hodnota tlaku spočítá jako součet p*, které je odhad pro řešení N-S rovnic a p´, který je určen pomocí tlakové korekce. Relaxační parametry α se mohou pro jednotlivé proměnné nastavit různě. Během výpočtu lze tyto hodnoty měnit a urychlit tak konvergenci výpočtu, v závislosti na velikosti reziduálů. Jsou-li reziduály velké, nastaví se malý relaxační faktor aby se utlumily nelinearity, pokud se změny reziduálů ustalují, může se relaxační faktor zvětšit. [5]

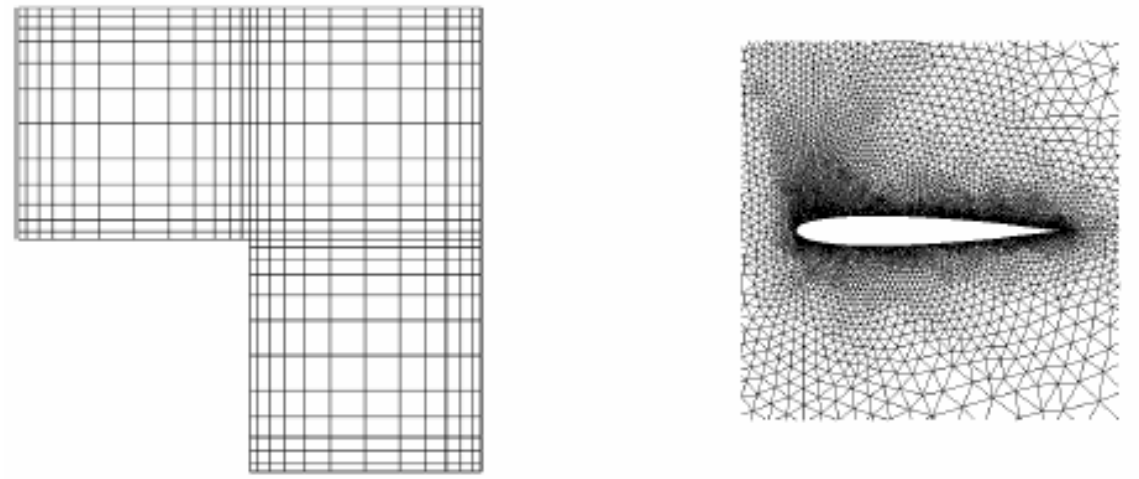
3.3 Diskretizační síť pro numerické výpočty

Diskretizační síť rozdělí modelovanou oblast na konečný počet elementů. Zásadním pravidlem je, že hrana nebo stěna elementů musí sousedit jen s jedinou hranou nebo stěnou sousedního elementu, nelze tedy libovolně zhušťovat síť. Počet elementů je přímo určující pro dobu výpočtu, s rostoucím počtem elementů roste velikost soustavy řešených rovnic. Doba výpočtu je také značně ovlivněna výpočetním

hardwarem, takže musíme přihlížet i tomuto aspektu. Kvalitní diskretizační síť se vyznačuje tím, že elementy jsou pravidelné a rovnoměrně rozprostřeny v celé řešené oblasti. Zároveň také elementy musí být dostatečné malé, aby například u turbulentního proudění zachytily mezní vrstvu.

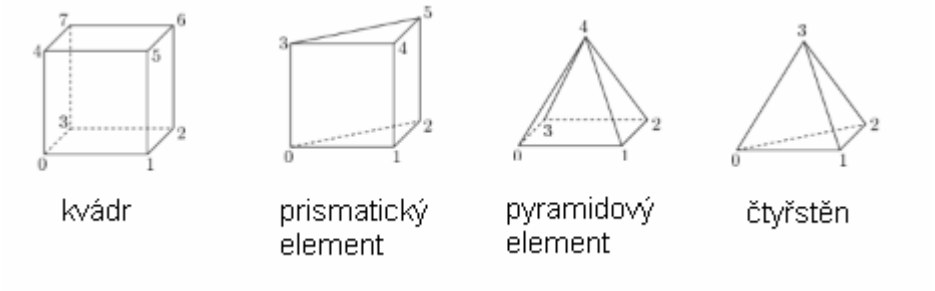
U reálných sítí nemůže splnit všechny požadavky na kvalitní síť, jelikož by byl počet elementů příliš vysoký a řešení takovéhoto problému neúměrně časově nákladné. To má za následek, že se poměrně často u sítí volí rozdílná velikostí elementů v řešené oblasti, výpočetní síť je lokálně zjemněna. Oblasti s hustší sítí se používají v důležitých místech, jakými jsou třeba oblasti s velkým gradientem rychlosti a změnou směru proudění tekutiny a také v mezních vrstvách. Zmenšování velikosti elementu by mělo být plynulé.

Nyní si představíme problém prostorové diskretizace pro metodu konečných objemů. Metoda konečných objemů je založena na rozdělení výpočetní oblasti na nepřekrývající se konečné objemy. Původně byla metoda odvozena pouze pro strukturované sítě, které se skládaly z uspořádaných kvádrů (viz obr. 3.4). To se postupem času ukázalo jako značně omezující pro geometricky složité oblasti.



Obr 3.4: Ukázka strukturované sítě nalevo a nestrukturované napravo

Tudíž bylo nutné upravit metodu konečných objemů natolik aby podporovala i výpočet na nestrukturovaných sítí (viz obr. 3.5).Ukázka nejběžnějších typů elementů používaných při výpočtech metodou konečných prvků jsou na obr. 3.5 .



Obr 3.5: Tvary konečných objemů [12]

4 Paralelní výpočty

4.1 Úvod do paralelního počítání

Jedním z důvodů, proč se začaly namísto sekvenčních výpočtů používat výpočty paralelní, bylo urychlení výpočtu. Spojením více výpočetních uzlů (počítačů) dohromady dostaneme vetší výkon než jen při použití jednoho počítače. Rozdělení výpočtu mezi více počítačů umožňuje řešení i velmi rozsáhlých úloh v poměrně rychlém čase.

Paralelní program se od sekvenčního programu liší v tom, že se při paralelním programu provádějí souběžně minimálně dvě aktivity (procesy). Vzájemná spolupráce procesů je zaručena komunikací např. pomocí zpráv. Obsahem zprávy může být výsledek mezivýpočtu nebo údaj o synchronizaci a jiné. Právě tato komunikace způsobuje, že s rostoucím počtem výpočetních uzlů se doba výpočtu nezkracuje úměrně počtu uzlů. Například při čtyřech výpočetních uzlech se výpočetní čas zkrátí méně než čtyřikrát. Proto správné naprogramování předávání zpráv má významný vliv na rychlost výpočtu. [3]

Dalším důvodem, proč paralelizací nedocílíme ideálního zkrácení výpočetního času, je ten, že ne celá část výpočtu jde paralelizovat, některé části výpočtu se musí řešit pouze sekvenčně. Příkladem je součet dvou mezivýsledků na jednotlivých výpočetních uzlech.

Paralelní výpočetní stroje se od běžných počítačů výrazněji neliší, obsahují stejné komponenty, ale liší se v jejich počtu a uspořádaní, některé typy paralelních výpočetních strojů se dokonce skládají z běžných počítačů. Jedno ze základních dělení paralelních počítačů je tzv. Flynova taxonomie. Ta dělí počítače podle počtu zároveň běžících instrukcí a datových toků. Dělení je následující:

- **SISD (Single Instruction, Single Data)**

Počítačová architektura s tímto názvem provádí pouze jednu instrukci nad jedním tokem dat. To odpovídá von Neumanově architektuře, která je základem všech počítačů s jedno jádrovýmprocesorem.

- **SIMD (Single Instruction, Multiple Data)**

Počítače založené na tomto principu provádí jednu operaci na více datech. Tato architektura se uplatňovala především v počátku rozvoje superpočítačů. Výhodou této architektury bylo rychlé zpracování vektorových operací. Nyní jí v počítačích zcela

nahradila MIMD. Naopak tuto architekturu nyní nalezneme ve všech grafických kartách, jelikož GPU (grafické procesorové jednotky) jsou tvořeny touto architekturou.

- **MISD (Multiple Instruction, Single Data)**

Architektura založená na tomto principu provádí více instrukcí nad jedinými daty. Málo využívaná architektura, v běžném životě se s ní nesetkáme. Místo, kde se tato architektura využívá je řízení vesmírných letů.

- **MIMD (Multiple Instruction, Multiple Data)**

Tato poslední architektura provádí více instrukcí nad více daty, na této architektuře je založena většina paralelních strojů od běžných vícejádrových procesorů až po výčetní clustery a superpočítače. Jelikož tato skupina je poměrně rozsáhlá, dělí se dále na počítače se sdílenou a distribuovanou pamětí. Počítače se sdílenou pamětí se vyznačují tím, že jsou všechny procesory propojeny pomocí sběrnice s centrální pamětí. Typickým zástupcem takovéto struktury je vícejádrový procesor. Naopak systémy s distribuovanou pamětí mají pro každý uzel vlastním paměť, jednotlivé uzly jsou spolu pak propojeny pomocí sběrnice a komunikace mezi jednotlivými uzly probíhá pomocí zpráv. Výhodou tohoto zapojení je spojení většího počtu výpočetních uzlů. [3]

Jelikož většina paralelních počítačů má architekturu MIMD s nesdílenou pamětí, došlo ještě k dalšímu dělení této skupiny dle [15]:

- **Symetrický multiprocesor SMP**

Jedná se o zapojení více stejných procesorů s jednou sdílenou paměti. Nevýhoda tohoto systému je v přístupu do společné paměti a z toho vyplývající omezení přenášeného množství dat šířkou sběrnice. To se řeší velkou vyrovnávací pamětí *cache*. Počet jader bývá výrazně omezen a může byt maximálně v řádu desítek. Velice drahé pro větší počet jader, výhodou je však rychlá komunikace mezi jednotlivými procesory oproti ostatním skupinám.

- **Masivně paralelní procesory MPP**

Jedná se o sestavu počítačů (procesorových uzlů) speciálně uzpůsobených pro vkládání do modulových skříní a propojených pomocí speciálních vysokorychlostních linek, které oproti clusteru zaručují rychlejší komunikaci mezi jednotlivými uzly. Výhodou je vysoká rychlost propojení jednotlivých uzlů, nevýhodou pak vysoká cena. Této způsob umožňuje zapojení řádově tisícovek výpočetních uzlů.

- **Cluster**

Velice podobný MPP jen s rozdílem, že komunikace mezi jednotlivými uzly je za pomoci LAN. Cluster má pomalejší komunikaci mezi jednotlivými uzly než MPP, na druhou stranu je finančně výhodnější.

- **Grid**

Grid je soustava výpočetních uzlů (domací počítače, malé clustery) rozmístěných i velmi daleko od sebe, propojených pomocí internetové sítě. Takovéto propojení může čítat i několik set tisíc uzlů. Nevýhodou je pomalá komunikace.

Speciální skupinu pak tvoří paralelní počítače složené z grafických procesorových jednotek (GPU– Graphic Processing Unit). Tyto procesory se postupně dali využit i k jiným účelům, než je jen práce s grafikou. Pro využívání GPU k obecným účelům se začala používat zkratka GPGPU (General-purpose computing on graphics processing units). Výhodou GPU je specializace na vektorové operace. To oproti univerzálním procesorům CPU vede k výraznému zrychlení výpočtu právě v oblasti maticových operací. Výhodou GPU je také mnohem rychlejší přístup do paměti díky velké datové prostupnosti sběrnice, navíc obsahují i několik set výpočetních jednotek na jednom čipu. Jejich nevýhodou je menší přesnost výpočtu oproti CPU jelikož datové typy mohou být maximálně 32 bitové.

Využití GPU k paralelním výpočtům je hlavně pro masivně paralelní úlohy. Tyto úlohy se vyznačují možností rozdělit výpočet na velký počet vláken, které mezi sebou jen velmi málo komunikují. Což mohou byt i CFD simulace. [16]

Nyní co jsme si udělali přehled paralelních strojů, si řekneme, jak se určuje výkon těchto zařízení. Výkon počítačů se udává ve FLOPS (Floating Point Operations Per Second), v českém překladu počet operací s plavoucí řádovou čárkou za sekundu. Toto hodnocení výkonu je však velice závislé na druhu počítané úlohy. Proto se musely sestavit takzvané testovací úlohy. Podle tohoto měřítka se hodnotí i nejvýkonnější počítače na světě (superpočítače). Nejvýkonnější počítač na světe je Tianhe-1A nacházející se v Číně, který byl uveden do provozu v roce 2010. Jeho výpočetní výkon je 2,566 petaFLOPS.a teoretický špičkový výkon je dokonce 4 až petaFLOPS. Tento počítač je vybaven 14336 procesory Xeon X5670 a 7168 GPGPU Nvidia Tesla M2050. [15]

4.3 Standardy (knihovny) pro psaní paralelních aplikací

Pro psaní paralelních programů jsou nejčastěji využívány dvě knihovny pro CPU a pak také speciální knihovny pro GPGPU. U CPU se jedná o knihovny MPI (Message Passing Interface) a PVM (Parallel Virtual Machine). Obdobou MPI je Open MPI určena pro open-source výpočetní balíky.

PVM se uplatňovala především v počátku vzniku paralelizace v 90.letech 20.století. Výhodou této knihovny je její jednoduchost, a to že dokáže pracovat i na uzlech, které se od sebe mohou velice lišit (architektura, velikost paměti, atd.).

Knihovna MPI je obdobou PVM a také pracuje na systému předávání zpráv. Tato metoda vznikla stejně jako PVM na začátku 90. let. Hlavní příčinou vytvoření této knihovny byla nutnost sjednocení a nahrazení jednotlivých knihoven, které vytvářel původně každý velký výrobce počítačů, univerzální knihovnu pro psaní paralelních programů. Nyní se tato knihovna stala takřka jedinou knihovnou pro psaní paralelních úloh, které se řeší na výpočetních clusterech a superpočítačích. Knihovna je založena na vzájemné komunikaci mezi jednotlivými uzly, dále také obstarává rozdělení úlohy na jednotlivé uzly a řídí přístup do paměti.

OpenMPI vznikl ze tři knihoven odvozených na základu MPI. Jedná se o knihovny vzniklé v akademickém prostředí. Důvodem vzniku této knihovny bylo vytvoření open-source knihovny poskytující vysoký výkon paralelizace a konkurenceschopnost vůči komerčním knihovnám.

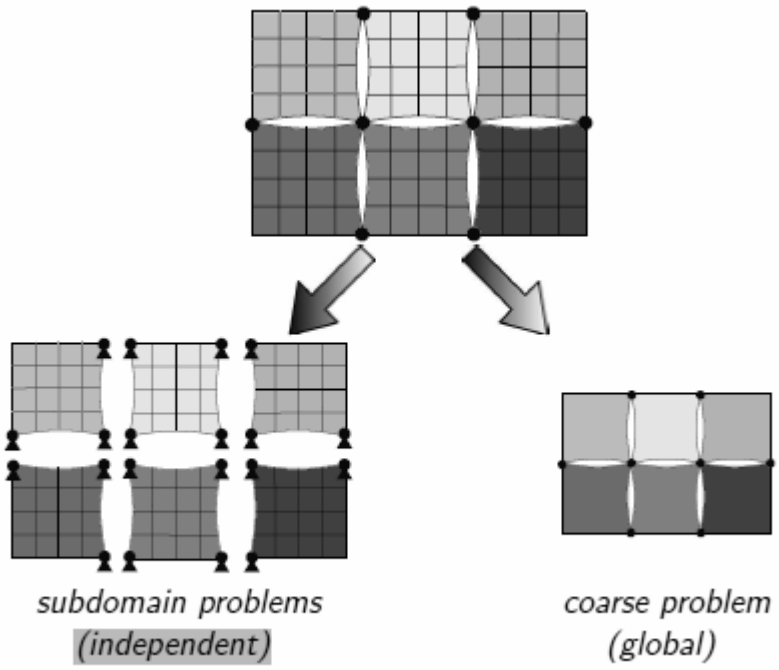
Nyní se krátce zmíníme o knihovnách pro GPGPU. Byl zaveden nový standart OpenCL, který podporuje paralelní výpočty jak na běžných procesorech CPU tak i na GPGPU. Než byl tento standart schválen a podporován výrobci hardwaru, zavedl si každý výrobce grafických karet svou vlastní knihovnu. V případě společnosti NVIDIA je to knihovna CUDA (Compute United Device Architecture). Tato knihovna je založena na programovacím jazyku C a umožňuje psaní programů jak pod Windows, tak pod Linux. V případě AMD to pak byla knihovna Brook+, která je stejně jako CUDA založena na programovacím jazyku C, ale není tak rozšířená, jelikož tato společnost v této oblasti neprorazila jako její konkurent NVIDIA. [16]

4.3 Dekomposzice oblasti

Dekomposice oblasti (Domain Decomposition) má za úkol rozdělit modelovanou oblast na soustavu podoblastí. Tyto podoblasti jsou při CFD jednotlivé části (objemy)

modelované oblasti. V těchto podoblastech se pak hledá výsledné proudové pole (rychlost, tlak, atd.). Jednotlivé podoblasti by na sobě měly být co nejvíce nezávisle. Řešení dané úlohy se pak hledá odděleně pro každou podoblast a nakonec se výsledky sloučí a vznikne výsledné řešení v celé oblasti. Provázanost jednotlivých podoblastí se pak hlídá pomocí tzv. hrubého prostoru, jedná se o řešení dané v celé modelované oblasti, každá podoblast je zastoupena jen několika málo uzly.

Metody dekompozice oblasti se dělí na dvě skupiny a to metody s překryvem a bez překryvu. Metody s překryvem se vyznačují tím, že průnik dvou sousedních podoblastí není prázdná množina, ale obsahuje uzly společné obou podoblastí. Do této skupiny metod patří Schwarzovy metody, které stály hlavně na počátku dekompozice oblasti. Postupem času byly nahrazeny právě metodami bez překryvu jako je metoda FETI, FETI-DP, BDDC. Jednotlivé podoblasti jsou spolu provázany pomocí rozhraní. V případě BDDC jsou tímto rozhraním uzly (viz obr. 4.1). [13]



Obr. 4.1: Dekompozice oblasti pomocí BDDC[18]

Metoda rozkladu FETI je duální metodou, která se používá pro řešení eliptických parciálních diferenciálních rovnic. Je určena pro řešení mechaniky pevných těles pomocí metody konečných prvků. Tato metoda však nebyla příliš vhodná pro rozsáhlé 3D oblasti, které se řešily na velkém množství výpočetních uzlů, a tak vznikla metoda FETI-DP. Jedná se o zjednodušenou a výkonnější verzi svého předchůdce FETI. Hlavní

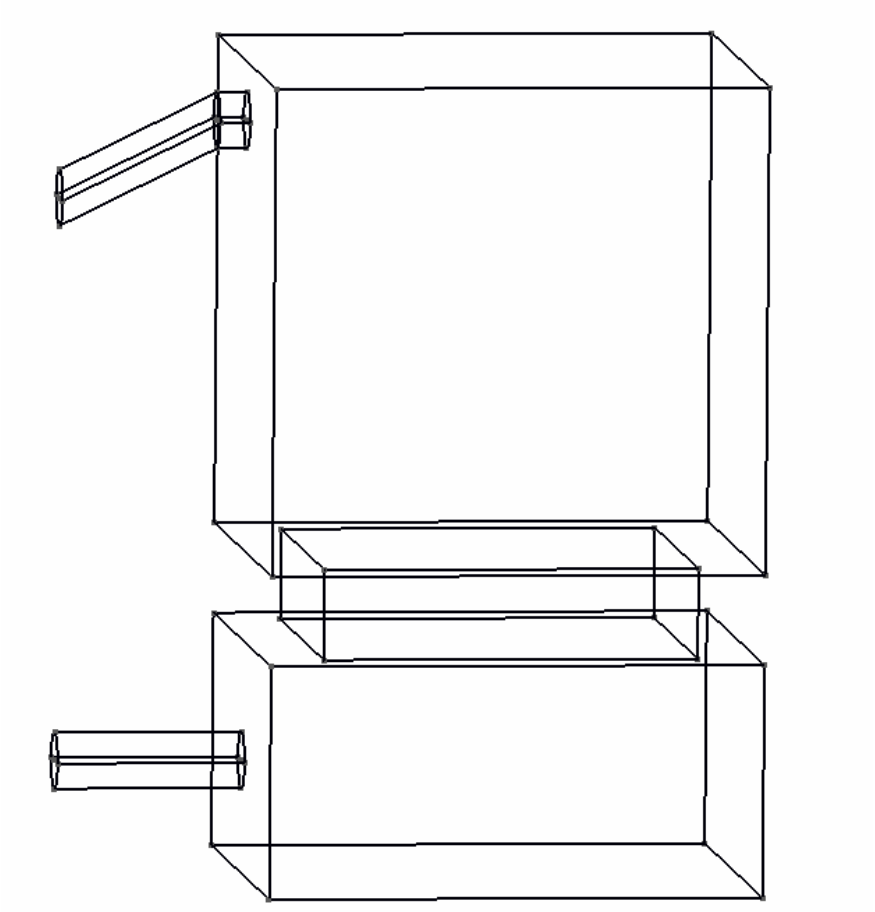
rozdíl je v zavedení Lagrangeových multiplikátorů na rozhraní podoblastí kromě rohů (vrcholových uzlů). Důležitým přínosem metody je, že se pomocí ní dají řešit úlohy čítající desetitisíce podoblastí.

Poslední metodou je metoda dekompozice oblasti BDDP, která je obdobou FETI-DP. Principem této metody je rozdělení řešené oblasti na podoblasti, které jsou spolu navzájem provázány pomocí hrubého prostoru. Hrubý prostor v tomto případě tvoří nejčastěji rohy podoblastí pro 2D úlohy nebo středy hran či středy stěn podoblastí pro 3D. Samotný výpočet je pak rozložen do dvou fází. V první se počítá řešení v jednotlivých podoblastí a v druhé pak globální řešení na hrubém prostoru. [18]

5 Numerické řešení proudění tekutiny pomocí výpočetního balíku OpenFOAM

5.1 Geometrie nádoby, generování sítě

Testy paralelních výpočtů byly provedeny na reálném případu proudění vzduchu přes filtrační kontejner. Výpočetní oblastí je tedy vnitřní objem kontejneru s částí vstupního a výstupního potrubí (viz obr 5.1). Filtr se v tomto případě do výpočtu nezahrnul.



Obr. 5.1: Geometrie nádoby pro testovaní účinnosti filtru na čištění spalin

Pro numerické výpočtu proudění bylo nutné vymodelovat geometrii výpočetní oblasti. Jelikož se jednalo o poměrně jednoduchou geometrii, nebylo jí nutné modelovat pomocí specializovaného CAD systému. Pro tyto účely byl použit program GMSH, který je zároveň generátorem sítí. Jedná se o open-source program, jehož autory jsou Christophe Geuzaine a Jean-François Remacle. Jeho největší výhodou je právě to, že se

jedná o open-source, dále umožňuje výběr síťovacího algoritmu, má dobře propracované uživatelské rozhraní, rozsáhlý a dobře zpracovaný manuál včetně příkladů a v neposlední řadě také podporuje více formátu pro ukládání dat. Tento program byl sice původně navržen výhradně pro akademické prostředí. Jeho asi největší nevýhodou je nekompatibilita s komerčními CAD systémy, které brání širšímu použití v průmyslových aplikacích. [11]

Geometrie výpočetní oblasti je zapsána v souboru pro pozdější úpravy, například při změně velikosti diskretizačního kroku sítě. Program GMSH obsahuje vlastní skriptovací jazyk, který umožňuje vstupy do tohoto programu zadávat pomocí textových souboru ASCII. Geometrii lze zadávat i v grafickém rozhraní.

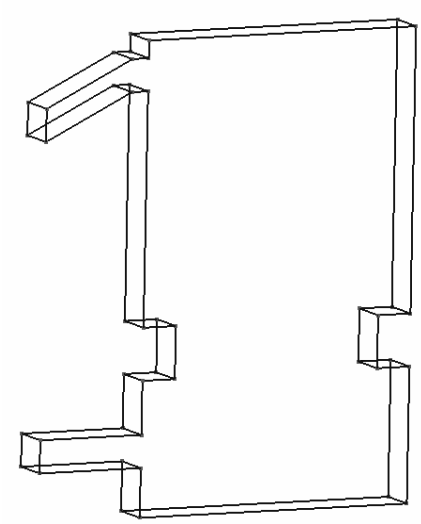
Geometrie reálného kontejneru na filtry byla pro účely výpočtu zjednodušena. Bylo nutné zanedbat šrouby, které slouží k upevnění a stabilizovaní filtru z důvodu chybného zasíťování v okolí těchto šroubů. Při výpočtech na těchto sítí docházelo k pádu výpočtu vlivem velké chyby výpočtu.

Pro účely paralelního výpočtu proudění bylo pomocí toho programu vytvořeno několik sítí o různém počtu elementů v závislosti na velikosti diskretizačního kroku. Jako síťovací algoritmus byla zvolena Delaunayova triangulace pro 3D, při které je oblast rozdělena na elementy typu čtyřstěn. Maximální počet elementů sítě vytvořené na geometrie nádoby byl řádově 3,5 mil. elementů, při větším počtu elementů byl program nestabilní a docházelo k jeho pádu. Generování sítě této velikosti bylo časově náročné, soubor obsahující tuto síť měl velikost přibližně 200MB.

Všechny sítě vygenerované sítě pomocí programu GMSH pro potřeby výpočtu jsou lokálně zjemněné. Jemnější síť je v obou potrubí, velikost elementů je zde poloviční oproti velikosti elementů ve zbylém objemu.

Abychom u programu GMSH nezmiňovali pouze výhody, byl během numerického výpočtu zjištěn poměrně závažný nedostatek u vygenerovaných sítí. Při výpočtu bylo zjištěno, že sítě vygenerované pomocí tohoto programu jsou v blízkosti některých hranic silně neortogonální, což mělo za následek znehodnocení výsledku výpočtu. Tento problém se dal vyřešit bud změnou geometrie nebo změnou výpočetního algoritmu. Změna geometrie spočívala v tom, že se použil 2D řez geometrie nádoby, který se poté pomocí funkce v GMSH extruduje (protáhne) do třetího rozměru (viz obr 5.2). Touto úpravou geometrie došlo k nahrazení původních čtyřstěnných elementů elementy typ kvádr s trojúhelníkovou podstavou. Toto změnou se však úplně změní

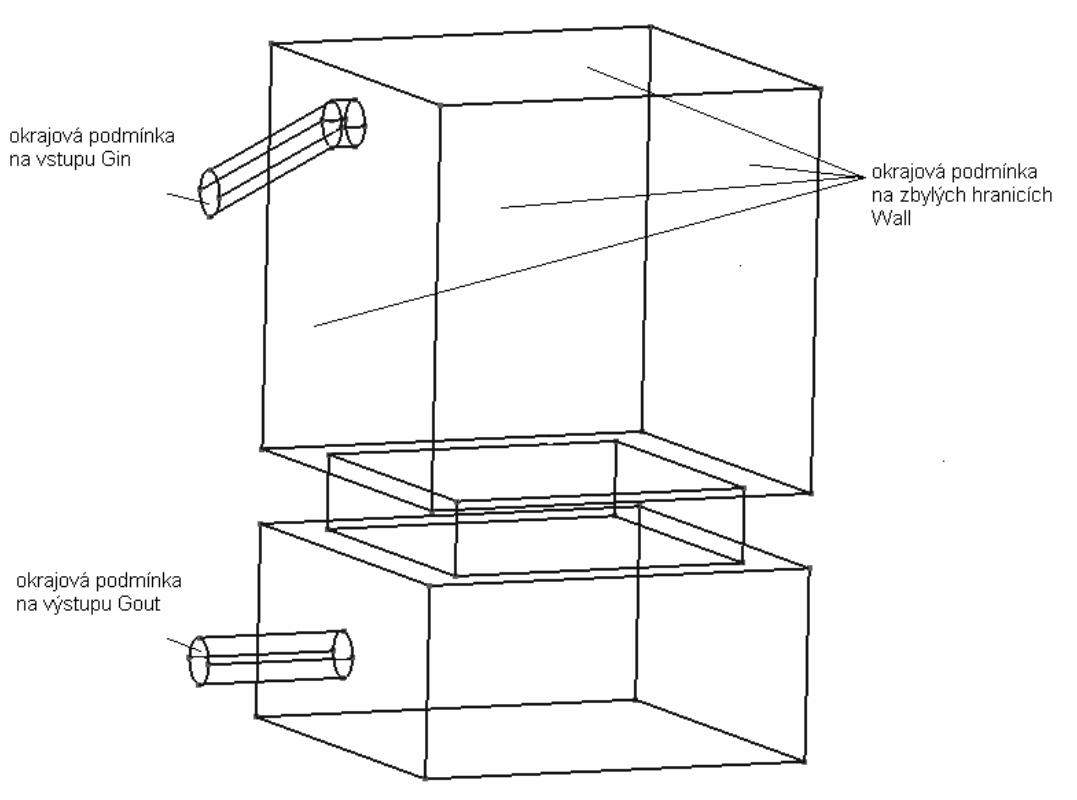
původní tvar geometrie, z potrubí se stanou kanály se obdélníkovým průřezem a výrazně se zvětší jejich velikost ve směru extruze.



Obr. 5.2: 2D řez geometrie nádoby

5.2 Okrajové a počáteční podmínky

Po vytvoření geometrie je nutné zadat okrajové a počáteční podmínky. Okrajové podmínky se zadávají pro všechny veličiny a na všechny hranice (viz Obr. 5.3). V našem případě, kdy se jedná o nestlačitelné laminární proudění, se zadávají okrajové podmínky pro rychlost a tlak. Pro tlak je zadána okrajová podmínka 1. druhu (Dirichletova) na výstupu (Gin), kde je předepsán nulový tlak, na zbylých hranicích je předepsána okrajová podmínka 2. druhu (Neumanova), nulová normálová derivace tlaku. Pro rychlost je pak zadána okrajová podmínka 1. druhu na vstupu Gin, kde je předepsána rychlost *u=0,11m/s* rozložená do složek (0.1 0 0.05)m/s, jelikož vstupní potrubí je pod úhlem 25°. Okrajová podmínka 1. druhu je také předepsána na stěnách (Wall), kde je předepsána nulová rychlost. Na výstupu je pak definována okrajová podmínka druhého typu, která udává nulovou normálovou derivaci. Pro přehlednost se vytvořila tabulka s okrajovými podmínkami pro jednotlivé hranice tab. 5.1. Soubory se zapsanými okrajovými podmínkami pro výpočetní balík OpenFOAM jsou v příloze A.



Obr. 5.3: Rozložení okrajových podmínek v geometrii nádoby.

Tab. 5.1: Okrajové podmínky pro úlohu proudění

hranice pro předepsání okrajové podmínky	okrajové podmínky pro rychlost u	okrajové podmínky pro tlak p
Gin	1. druhu (0,1 0 0,05)	2. druhu (nulová normálová derivace)
Gout	2. druhu (nulová normálová derivace)	1. druhu (0 0 0)
Wall	1. druhu (0 0 0)	2. druhu(nulová normálová derivace)

Po zadání okrajových podmínek se zadají podmínky počáteční. Pro tlak se v celém objemu v čase nula předepíše nulová hodnota. U rychlosti se pak v čase nula předepíše také nulová hodnota rychlosti kromě oblasti vstupu, kde se předepíše rychlost (0,1 0 0,05)m/s.

5.3 Numerické řešení úlohy pomocí výpočetního balíku OpenFOAM

Výpočetní balík OpenFOAM je založen na metodě konečných objemů. Jedná se open-source program, který je celý psán v programovacím jazyce C++. Uživatel má možnost zasahovat do vnitřní struktury programu a tím si ho přizpůsobit pro jistý specifický problém. OpenFOAM tvoří ucelený balík, který umožňuje danou úlohu zpracovat komplexně bez použití jiného softwaru, jelikož obsahuje nástroje pro modelování geometrie a generování konečně objemových sítí, celou škálu řešičů pro různé fyzikální problémy od výpočtu stlačitelného a nestlačitelného proudění tekutiny, mechaniku pevných těles, elektromagnetismu až po finance. K zobrazení vypočtených výsledků slouží vizualizační programu ParaView, který je propojen s balíkem OpenFOAM.

Vstupy jsou zadávány pomocí textových souborů, z důvodu univerzálnosti tohoto programu. Tyto soubory definují vše od geometrie, přes výběr solveru a jeho parametrů až po zadávání okrajových podmínek. Z tohoto důvodu je nutné před samotným výpočtem všechny tyto soubory vytvořit, aby bylo možné výpočet spustit. Nejdříve se musí vytvořit soubor obsahující geometrii modelové oblasti, na které se následně vygeneruje diskretizační síť. Zde výpočetní balík OpenFOAM nabízí několik možností. První možností je využít vlastního generátoru sítí BLOCKMESH nebo se může použít jiný software a data pomocí konverortu přeformátovat pro potřeby OpenFOAMu. Konvertor podporuje formáty běžně používaných softwarů. Mezi tyto softwary patří ANSYS, Fluent, Gambit, a další bližší informace k této problematice jsou popsány ve [12]. V našem případě byl zvolen druhý způsob a geometrie i síť byly vytvořeny v externím programu GMSH a následně překonvertovány. A to z toho důvodu, že pro naší geometrii kontejneru by bylo velice komplikované měnit hustotu sítě. Tato činnost totiž vyžaduje měnění velkého množství hodnot v soboru s geometrií. Naproti tomu v GMSH se to provede pouze změnou jediného parametru. Druhým nedostatkem generování sítí pomocí OpenFOAMu je, že se geometrie před zasíťováním musí rozdělit na bloky tak, aby byla pokryta celá modelová oblast a jednotlivé bloky na sebe navazovaly celými stěnami a hranami. To by mělo za následek problematické spojení vnitřního objemu nádoby se vstupním a výstupním potrubím, protože potrubí je kruhového průřezu. [12]

Pro numerické řešení proudění vzduchu v kontejneru musíme popsat danou úlohu matematickým modelem. Obsahem této práce je řešení úlohy proudění vzduchu v nádobě na testování účinnosti filtru. Jelikož je zadána malá rychlost na vstupu

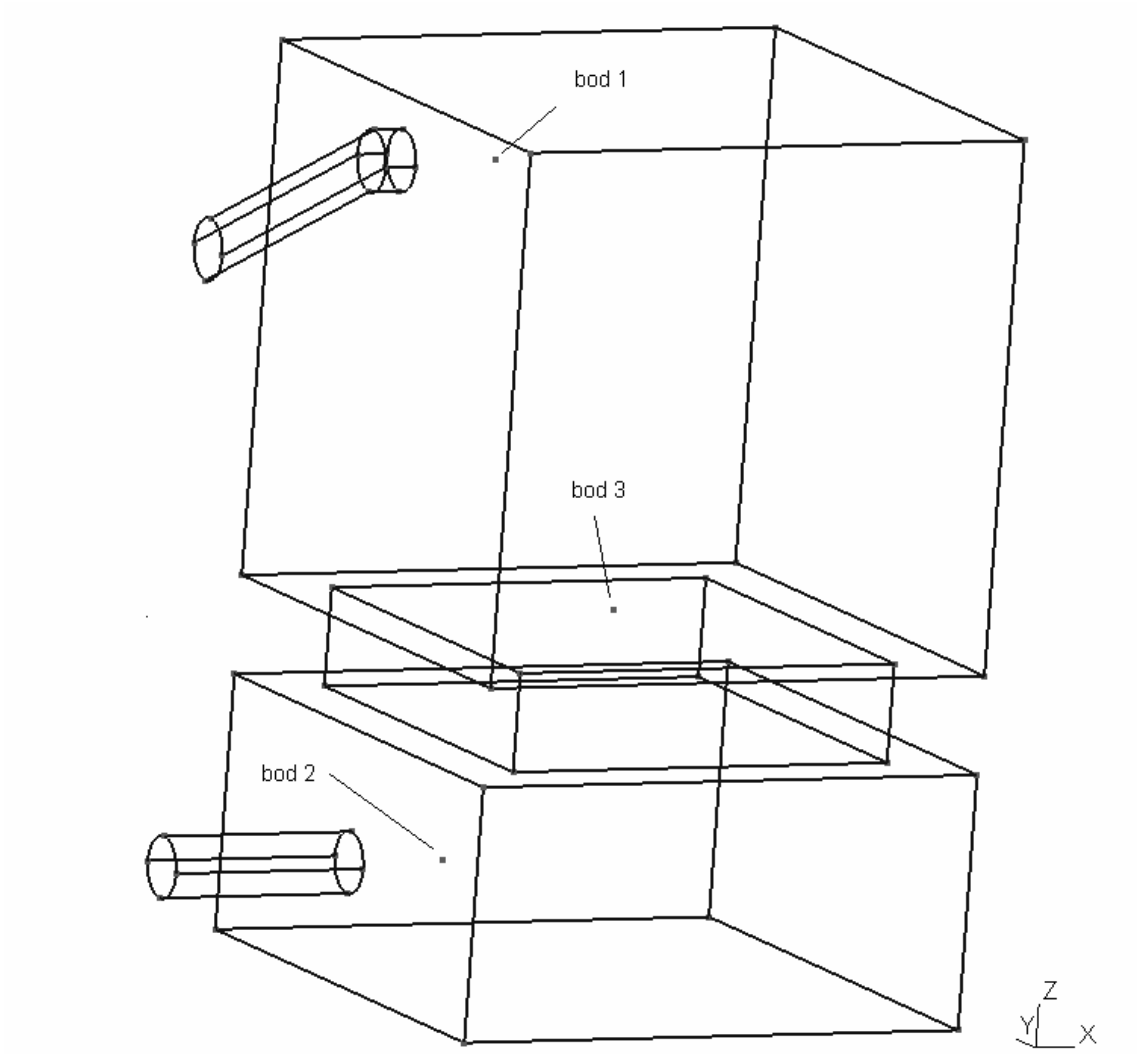
u=0,11m/s a tedy malá Re, jedná se o proudění laminární. Další zjednodušující předpoklad, který se může zavést díky nízké rychlosti, je ten, že se proudění plynu dá modelovat jako nestlačitelné, jelikož je splněna podmínka *u<0,3Ma*. Proudění je pak popsáno rovnicemi kontinuity a Navier-Stokesovými rovnicemi pro nestlačitelné proudění:

$$\nabla \cdot u = 0$$
$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u - \nu \Delta u + \frac{1}{\rho}\nabla p = 0 \quad . \tag{5.1}$$

Před samotným výpočtem musíme vybrat řešič, který využijeme pro výpočet daného problému. V případě modelování proudění je na výběr z několika řešičů, jak pro stlačitelné tak nestlačitelné proudění. Celkový přehled řešičů je v [12]. Nakonec byl v tomto případě vybrán řešič SIMPLE, který počítá ustálený stav proudění. Parametry tohoto řešiče jsou počet iterací a relaxační koeficienty pro jednotlivé veličiny (tlak a rychlost). Relaxační koeficienty byly zvoleny podle příkladu v tutoriálu, pro tlak tedy byla nastavena hodnota relaxačního koeficientu na *0,3* a pro rychlost *0,7*. Po nastavení relaxačních koeficientů bylo nutné zvolit vhodný počet iterací tak, aby se ustálily hodnoty veličin a zároveň, aby se nepočítalo zbytečně, když už jsou veličiny ustálené. Pro tento účel byly v obejmu nádoby zvoleny body, ve kterých se sledovala hodnota tlaku a velikost rychlosti(viz Obr. 5.5 a Tab. 5.2).

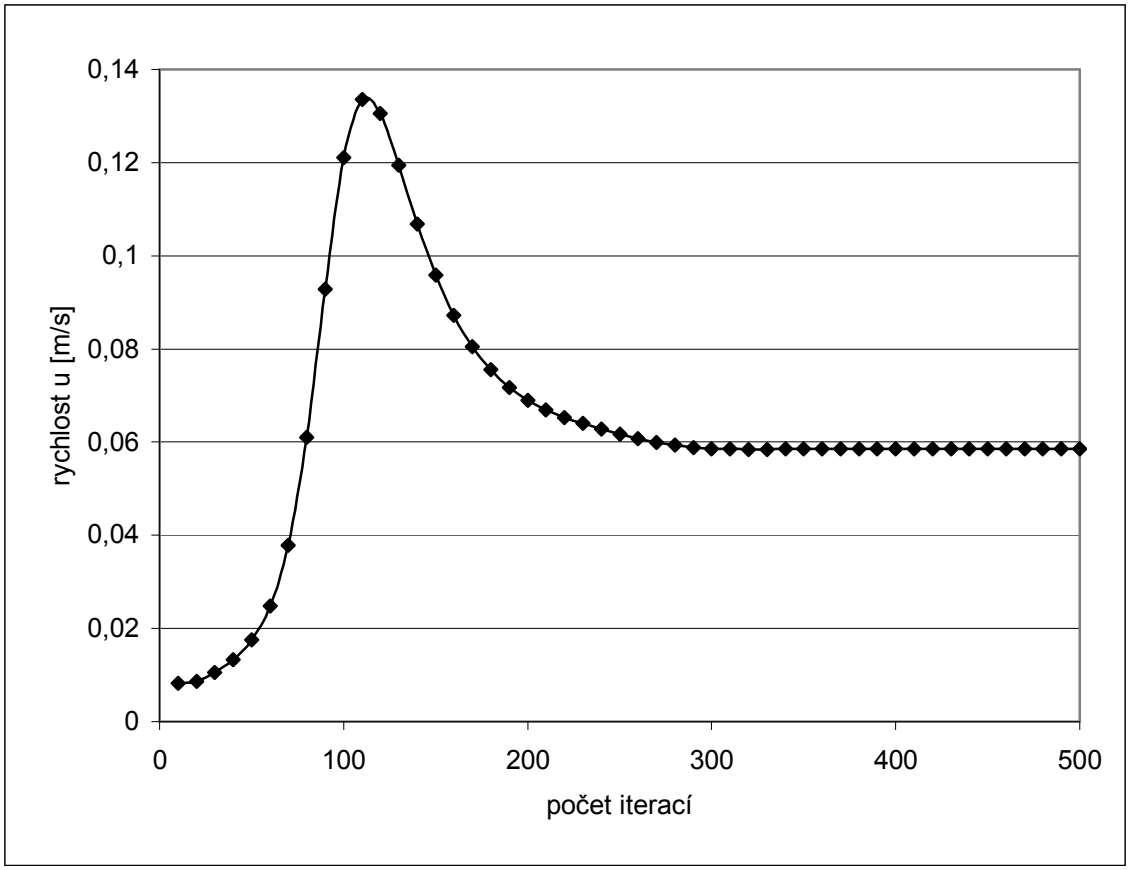
Tab. 5.2: Souřadnice bodů ve kterých se zaznamenával tlak a rychlost

	souřadnice x [m]	souřadnice y [m]	souřadnice z [m]
bod 1	0,025	0,066	0,2
bod 2	0,025	0,066	0,03
bod 3	0,066	0,066	0,09

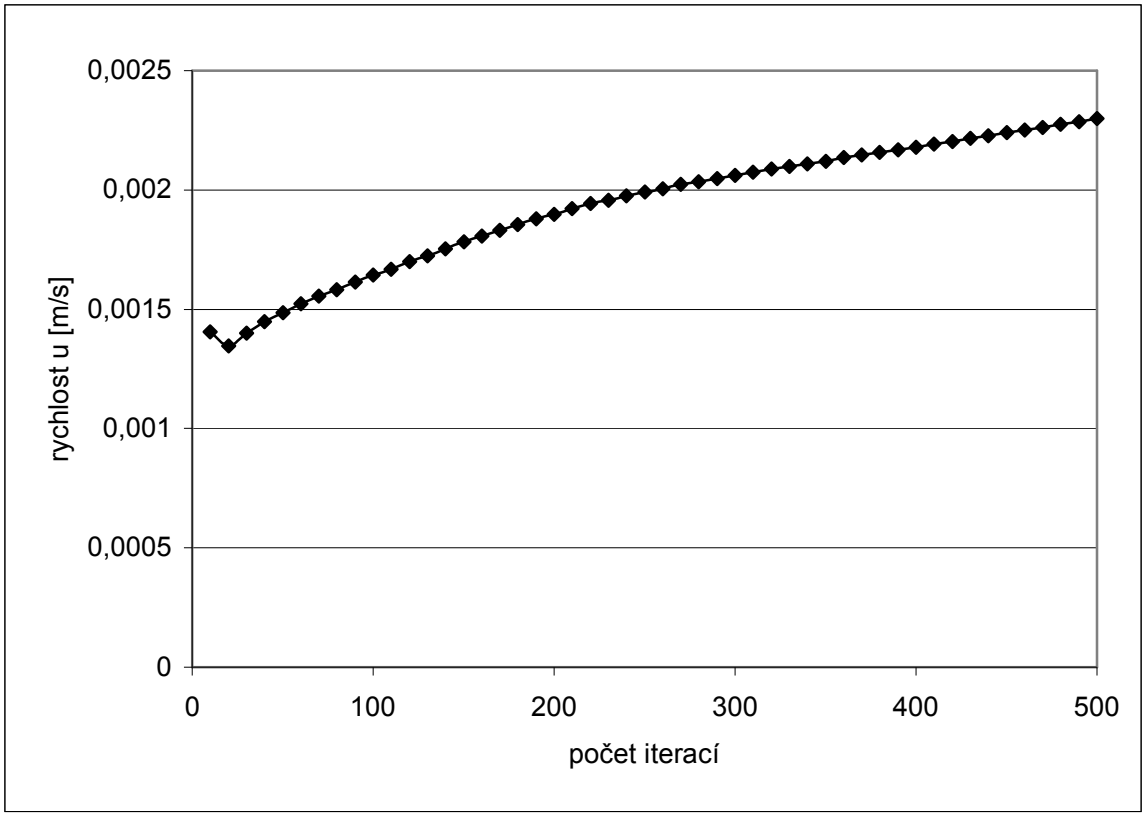


Obr. 5.5: Rozmístění kontrolních bodů v objemu nádoby

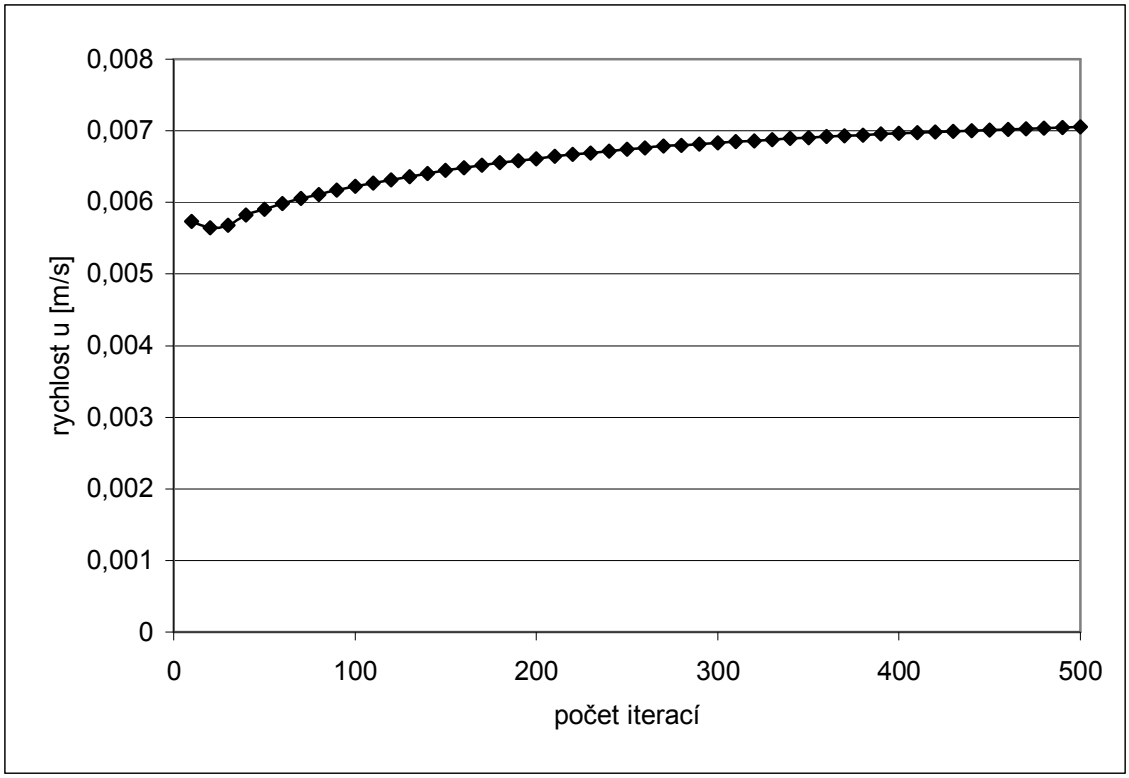
Hodnoty tlaku a rychlosti byly vyneseny do grafů, aby bylo jasně patrné, jak se hodnoty proměnných ustalují v závislosti na počtu iterací. Na prvních třech grafech je zaznamenán vývoj rychlosti v jednotlivých kontrolních bodech. V kontrolním bodu 1 (viz graf 5.1) a v bodu 2 (viz. graf 5.3) se hodnota rychlosti ustálí během 500 iteracích, naproti tomu v bodu 3 (viz graf 5.2) se velikost rychlosti se neustálí ani po 500 iteracích. Z toho vyplývá, že počet iterací by měl být zvětšen. Tlak se ustálí po 400 iteracích (viz graf 5.4, graf 5.5 a graf 5.6). Hodnota tlaku se ve všech bodech ustaluje stejně (stejný průběh). Velikost tlaku je pak pro každý bod jiná, ale rozdíl je velice malý proto se hodnota jeví jako shodná.



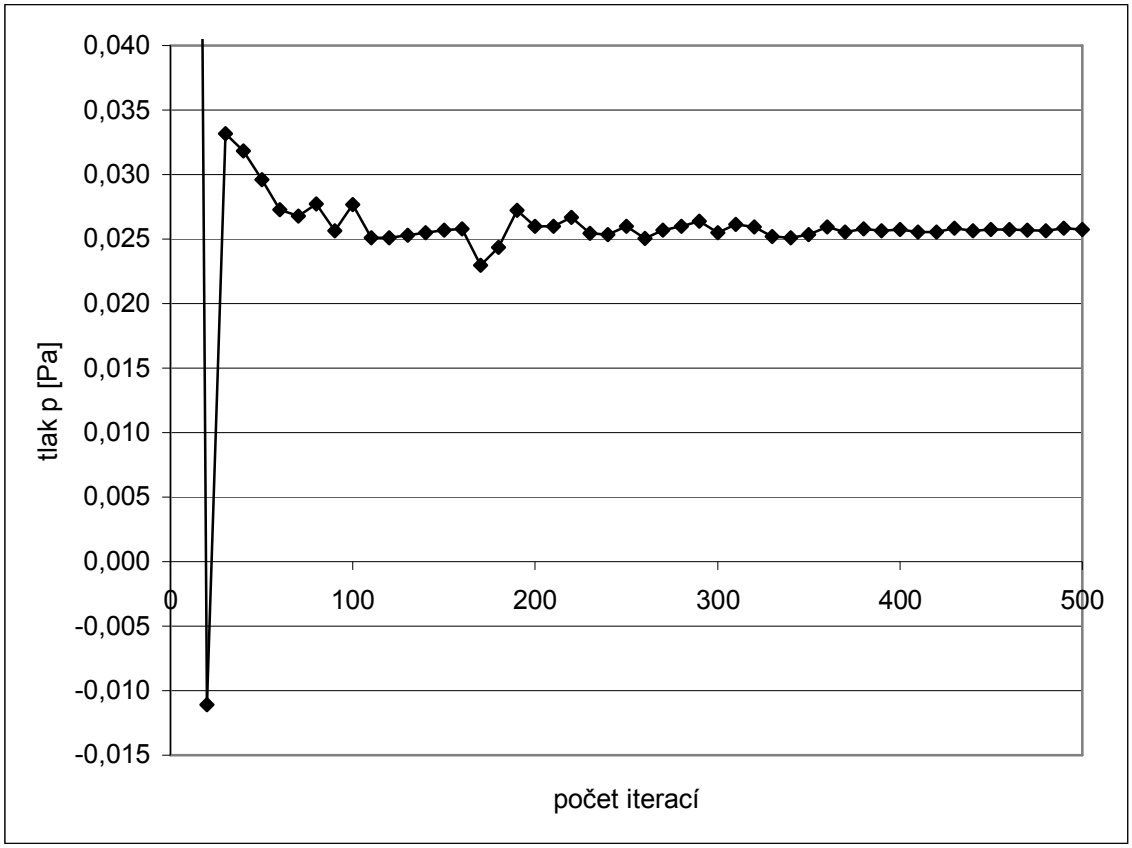
Graf 5.1: Vývoj hodnoty rychlosti v bodu 1



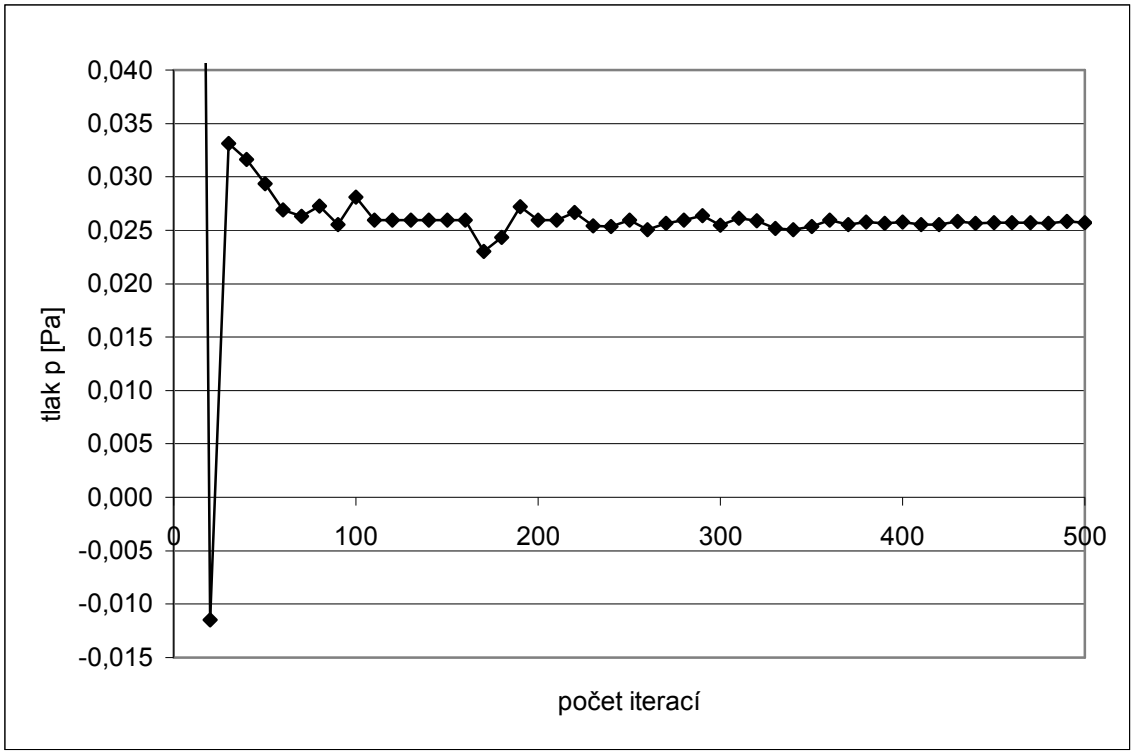
Graf 5.2: Vývoj hodnoty rychlosti v bodu 3



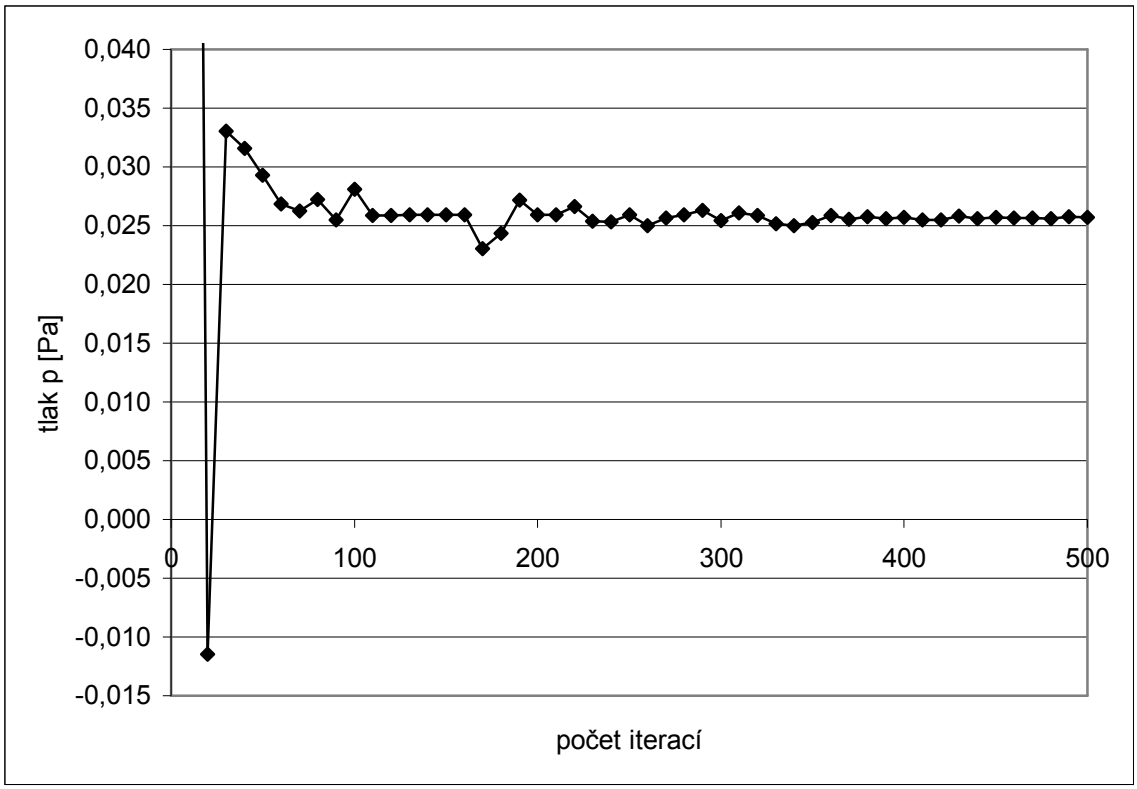
Graf 5.3: Vývoj hodnoty rychlosti v bodu 2



Graf 5.4: Vývoj hodnoty tlaku v bodu 1

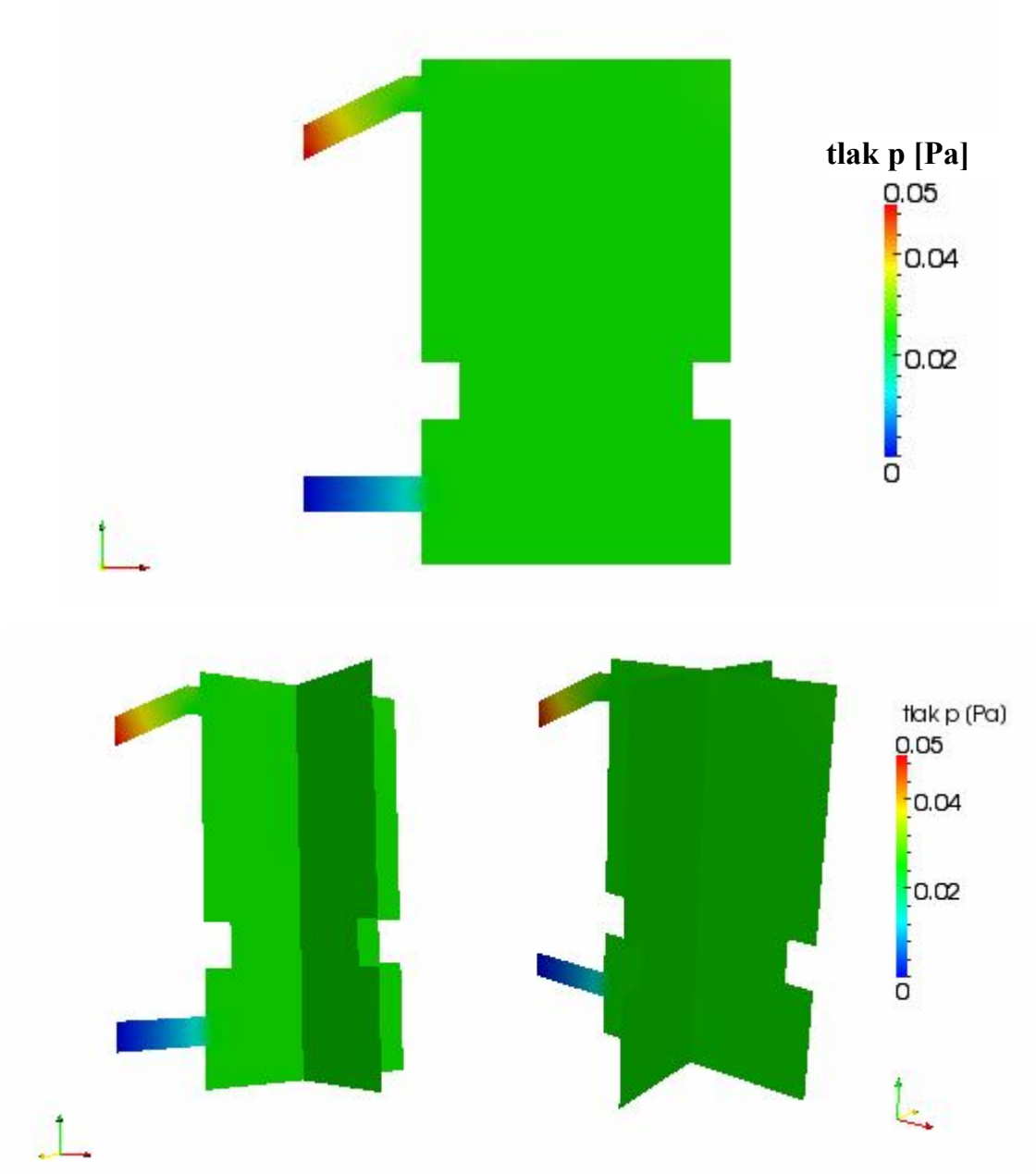


Graf 5.5: Vývoj hodnoty tlaku v bodu 3



Graf 5.6: Vývoj hodnoty tlaku v bodu 2

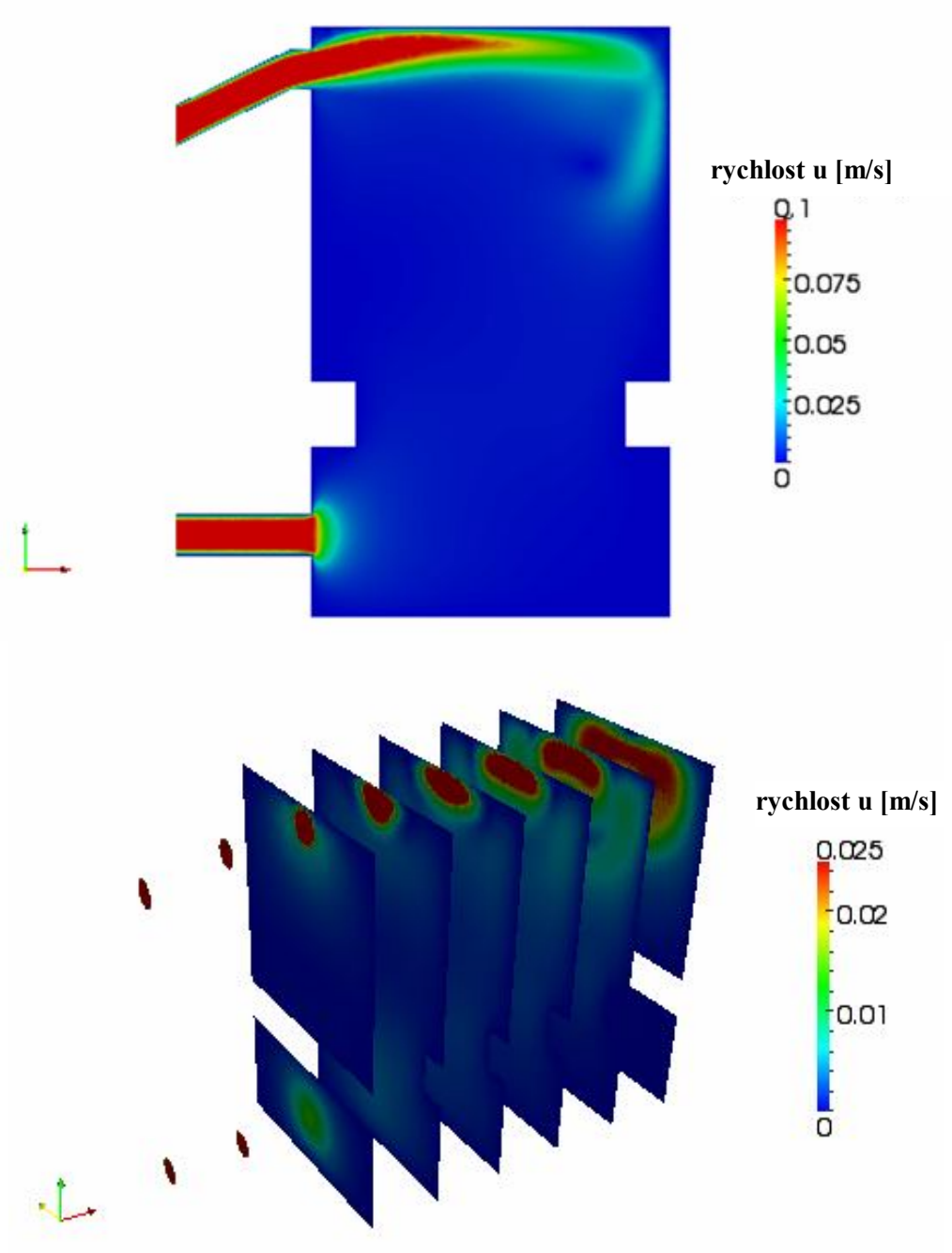
Vizualizace vypočtených výsledků byla provedena pomocí programu ParaView. Tímto nástrojem lze vytvářet jak statické obrázky, tak animace. V nabídce je celá paleta nástrojů pro vizualizaci výsledků. Pomocí tohoto programu byly zobrazeny výsledné tlakové a rychlostní pole po výpočtu 500 iterací.



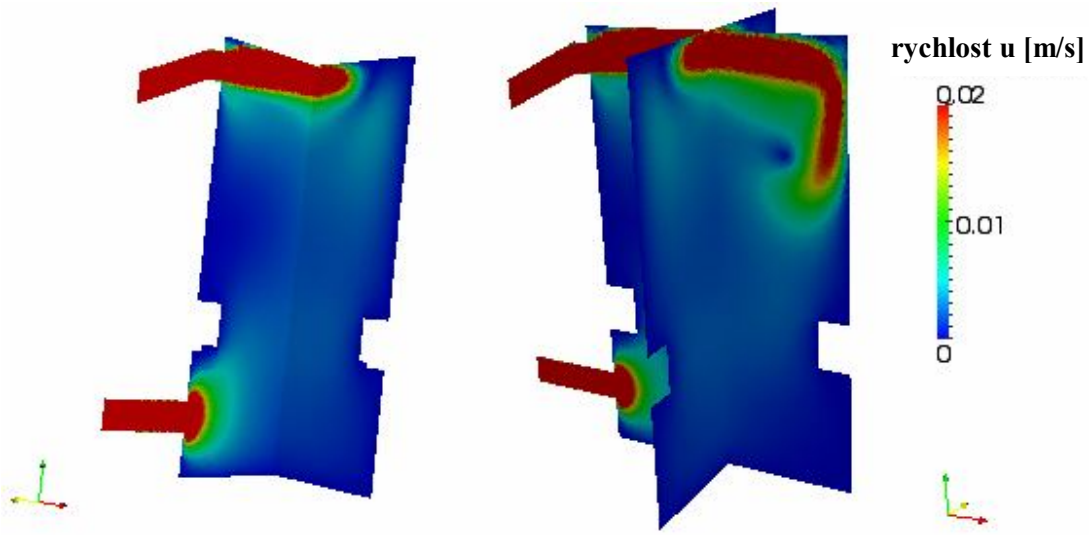
Obr. 5.6: Tlakové pole po výpočtu 500 iterací, rychlost na vstupu u=0.1 m/s

Pro tlak byl vyobrazen pouze řez středem nádoby a křížový řez (viz obr. 5.6). Jelikož se jedná o nestlačitelné proudění a rychlost proudění je velmi nízká, nevznikají zde žádné víry ani úplavy, které by hodnotu tlaku v objemu ovlivňovaly. Naproti tomu u rychlosti pouhé vyobrazení středového řezu nestačí. Pro lepší přehlednost bylo

vytvořeno několik řezů kolmo na středový řez (řezy ve směru osy y) (viz obr. 5.7) a poté taky křížový řez (viz obr. 5.8). Z důvodu větší přehlednosti byly řezy v obr. 5.7 od sebe posunuty, proto výsledek vypadá, jako by byla modelovaná oblast protažena ve směru osy y.

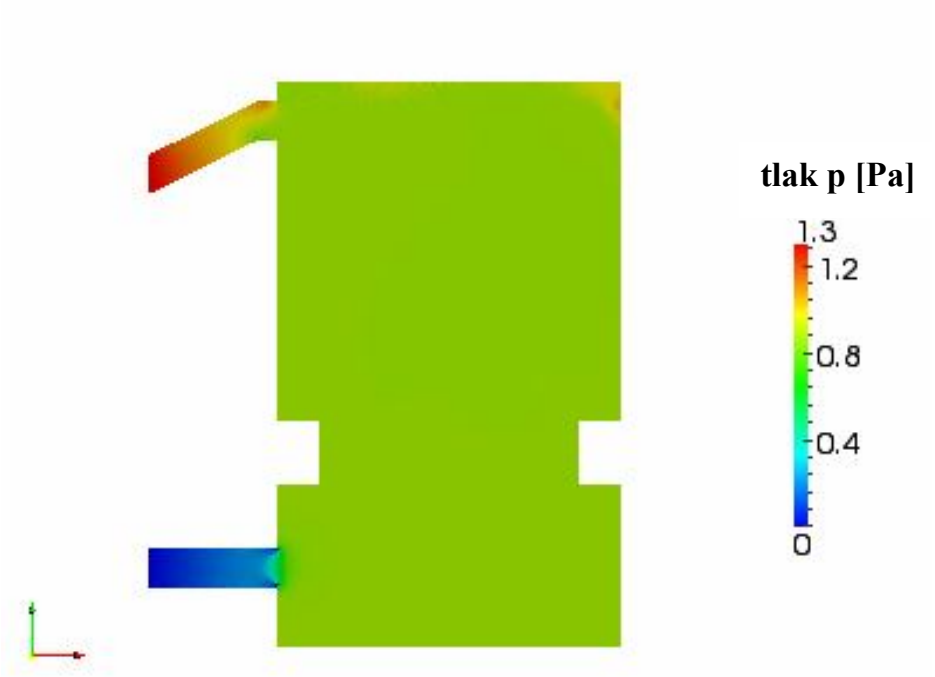


Obr. 5.7: Rychlostní pole po výpočtu 500 iterací, rychlost na vstupu u=0.1 m/s

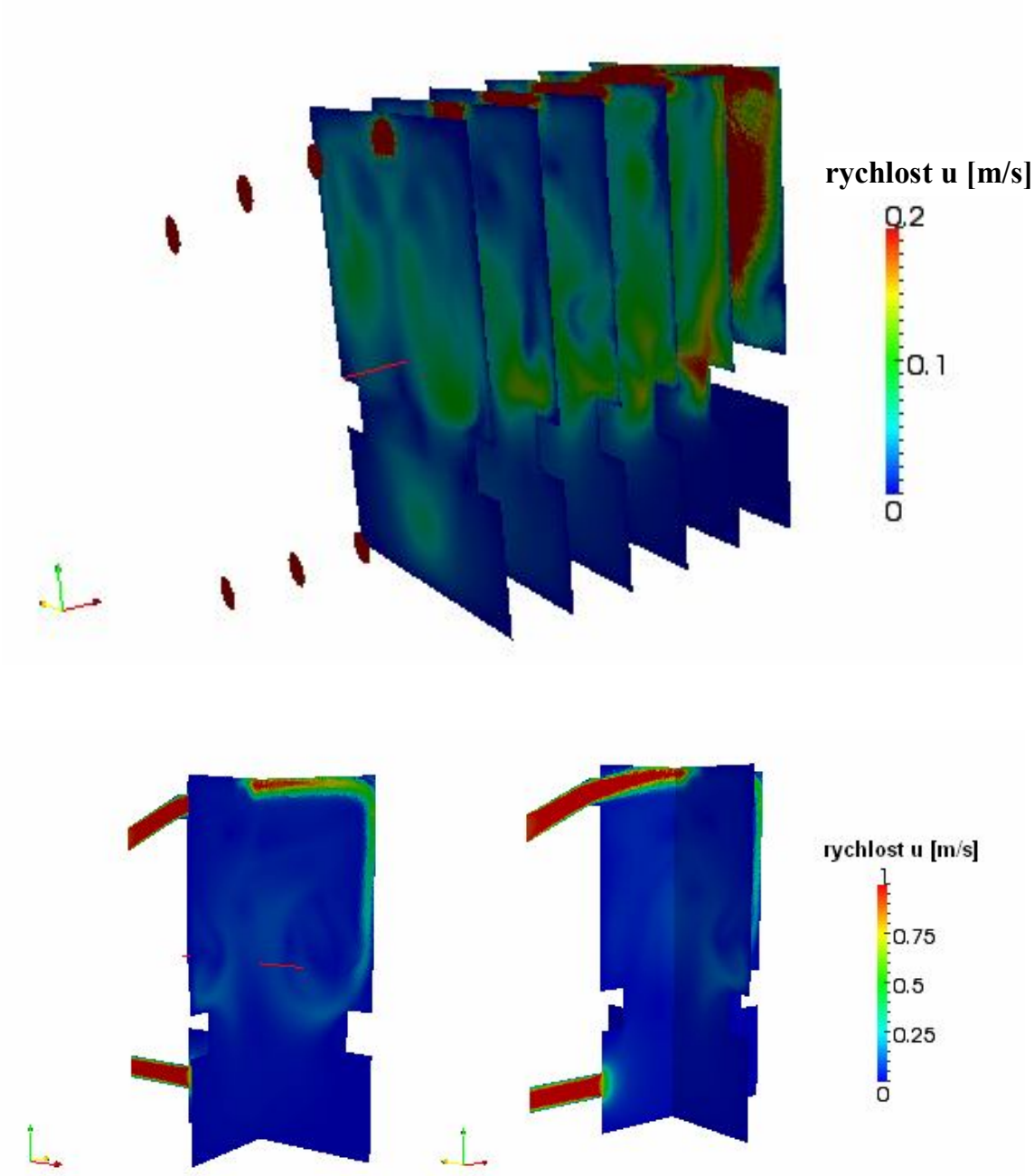


Obr. 5.8: Rychlostní pole po výpočtu 500 iterací (křížový řez), rychlost na vstupu u=0.1 m/s

V rámci diplomové práce byl také proveden výpočet při vyšší vstupní počáteční rychlosti, kdy *u=1m/s* a opět se sledovalo výsledné pole rychlosti (viz obr. 5.10) a tlakové pole (viz obr. 5.9) Oproti nízké vstupní rychlosti z předchozích obrázků, je zde již patrné, že se začínají projevovat poruchy proudění vlivem přechodu k turbulentnímu proudění a výsledné pole se ani po 500 iterací neustálí. Bylo by nutné namísto matematického modelu pro laminární proudění zvolit matematický model pro turbulentní proudění.



Obr. 5.9: Tlakové pole po výpočtu 500 iterací, rychlost na vstupu u=1 m/s



Obr. 5.10: Rychlostní pole po výpočtu 500 iterací, rychlost na vstupu u=1 m/s

6 Paralelní výpočty pomocí výpočetního balíku OpenFOAM a jejich testování

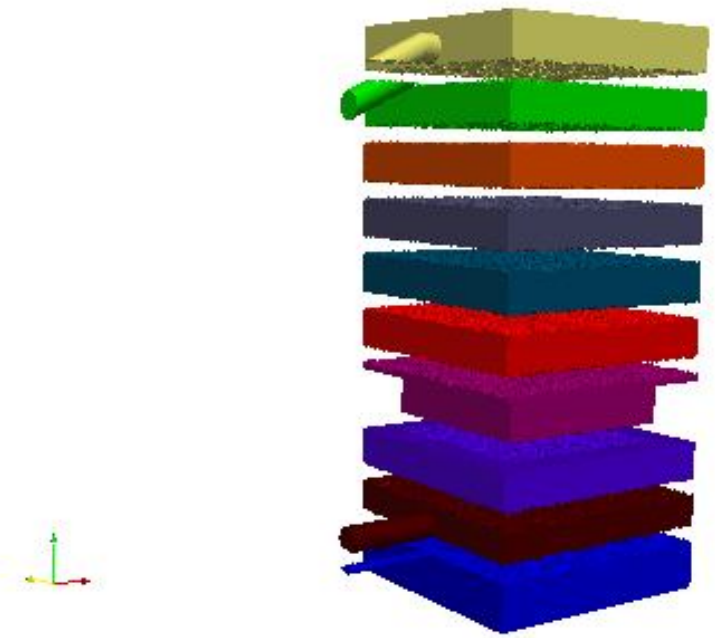
6.1 Paralelní výpočty pomocí výpočetního balíku OpenFOAM

Výpočetní balík OpenFoam je schopen mimo sekvenčních výpočtů vykonávat i výpočty paralelní. Oproti sekvenčnímu výpočtu je řešení úlohy proudění pomocí paralelního výpočtu složitější. K paralelním výpočtům je potřeba vhodný hardware (vícejádrový, víceprocesorový systém, cluster nebo superpočítač), paralelní algoritmy a nakonec je nutné také provést rozklad oblasti (dekompozice oblasti). Výpočetní balík OpenFOAM nabízí celou řadu paralelních solverů, pro řešení nestlačitelného proudění, opět byl vybrán řešič SIMPLE. Dalším krokem je tedy rozklad oblasti (dekompozice), touto úpravou se rozumí rozdělení diskretizační sítě a k tomu příslušných okrajových a počátečních podmínek na podoblasti (subdomény). Rozklad oblasti musí danou síť rozdělit tak, aby bylo rozdělení rychlé a zároveň umožňovalo efektivní paralelní výpočet. Každá podoblast se pak řeší na jednom procesoru (jádru), komunikace informací na společných hranicích probíhá pomocí MPI (message passing interface). Po skončení výpočtu je nutné výsledky zrekonstruovat (složit), abychom získali výsledná pole rychlosti a tlaku.

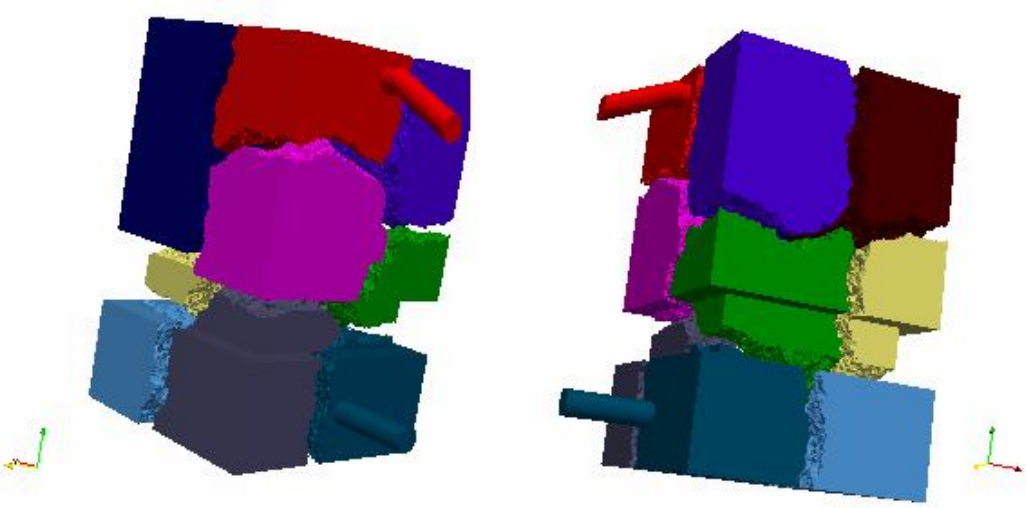
Pro rozklad oblasti jsou v OpenFOAMu naprogramovány čtyři metody. Těmito metodami jsou *SIMPLE*, *HIERARCHICAL, METIS, SCOTCH* a *manual*. Metoda SIMPLE pracuje na jednoduchém principu, a to tak, že danou oblast rozdělí na celky, o stejném počtu elementů (viz obr. 6.1). Způsob dělení určuje uživatel, který zadává na kolik dílů se daná oblast bude v jednotlivých směrech souřadných os dělit. Další metodou je metoda *HIERARCHICAL*, která je obdobou metody *SIMPLE* jen s tím rozdílem, že si uživatel určuje pořadí směrů dělení.

Výrazně odlišnou metodou rozkladu oproti předešlým metodám je metoda *METIS*. Princip metody METIS je ten, že danou oblast rozdělí tak, aby byla plocha hranic mezi jednotlivými podoblastmi co nejmenší, snaha co nejvíc minimalizovat komunikaci při výpočtu (viz obr. 6.2). Navíc tato metoda umožňuje přidělovat jednotlivým jádrům váhový koeficient, toho se dá využít, pokud máme k dispozici výpočetní zařízení složené z různých typů procesorů. Tento váhový koeficient určuje, jak velká část úlohy na dané jádro připadne v závislosti na jeho výpočetním výkonu. Toto nastavení provádí sám uživatel, proto je nutné předem znát výkon jednotlivých výpočetních jednotek nebo

použít stejné procesory a těm nastavit stejný váhový koeficient. Algoritmy realizován v METIS jsou založeny na víceúrovňovém rekurzivní-půlení intervalu, a multi-omezené dělení systému. Obdobou metody *METIS* je metoda *SCOTCH* tato metoda pracuje na stejném principu jako METIS a je dostupná open-source verze OpenFOAMu, zatím co *METIS* je pouze pro komerční verzi.



Obr. 6.1: Výsledný rozklad oblasti na 10 podoblastí pomocí metody SIMPLE



Obr. 6.2: Výsledný rozklad oblasti na 10 podoblastí pomocí metody METIS

Poslední metodou je *manual,* i když se vlastně nejedná přímo o metodu, protože uživatel sám provede rozdělení oblasti. Dekompozice oblasti na 10 podoblastí pomocí metod METIS a SIMPLE je demonstrována na obr. 6.2, zdrojové soubory jsou v přílohách B a C.

Po provedení rozkladu oblasti jsou výsledky této operace zapsány do souborů, každá podoblast zvlášť. [12] Po provedení dekompozice se musíme specifikovat hardware (přesný název v síti a počet jader), na kterém se provádí výpočty. Toho docílíme vytvořením souboru, do kterého zapíšeme přesné názvy jednotlivých výpočetních uzlů a uvedeme počet jader pro každý uzel. Soubor se seznamem uzlů je k nahlédnutí v příloze D.

Paralelní výpočty se realizují na školní výpočetní cluster Hydra. Jedná se o smíšený cluster složeny z dvou typu výpočetních uzlů a to Intel XEON a AMD Opteron (viz. tab. 6.1). Uzly Intel XEON obsahují dva dvoujádrové procesory zatímco uzly AMD Opteron obsahují dva jednojádrové procesory. Nevýhoda clusterů všeobecně je v propojování uzlů, zde dochází k největšímu

Tab. 6.1: Parametry výpočetního clusteru Hydra

12 uzlů Dell PowerEdge 1950
2x Intel Xeon 5140 2.33GHz/4MB 1333FSB (2 jádra)
4GB RAM 667MHz (4x1HB)
80GB SATA2, 7200 ot./min, hot plug
2x NIC 1 Gbps, Broadcom NetXtreme II 5708 Gigabit Ethernet NIC
DVD ROM
Celkem: 24 CPU, 48 jader, 48GB RAM, 960GB HDD
17 uzlů Sun Fire V20z (1U)
2x AMD Opteron 252, 2600 MHz (1 jádro)
4 GB RAM
73 GB HDD, 10025 ot./min, Fujitsu MAT3073NC
1x Dual Ultra320 SCSI, LSI Logic 53c1030 PCI-X
2x NIC 1 Gbps, Broadcom BCM5704
DVD-ROM, FDD
Celkem: 34 CPU (34 jader), 68 GB RAM, 1.2 TB HDD

6.2 Parametry pro testovaní paralelních výpočtů

Aby bylo možné paralelní výpočty porovnávat, existují parametry, podle kterých se hodnotí efektivita paralelního výpočtu. V závislosti na těchto parametrech se určuje, zda je úloha dobře paralelizovatelná, či nikoliv. V našem případě navíc pak poslouží k určení, která metoda rozkladu oblasti a který výpočetní hardware je vhodnější použít pro paralelní výpočty naší úlohy.

Prvním parametrem je zrychlení paralelního výpočtu, které je definováno vztahem

$$S_p = \frac{T_s}{T_p} \quad , \tag{6.1}$$

kde T_s je výpočetní čas sekvenčního výpočtu, T_p je výpočetní čas paralelního výpočtu a index p udává počet jader. Paralelní výpočetní čas je součtem času potřebného k režii a samotným časem pro vykonání výpočtu přidělené úlohy. Zrychlení nám udává, jak moc se výpočet zrychlí při použití paralelního výpočtu oproti sekvenčnímu výpočtu. Ideální zrychlení je rovno počtu jader a nazývá se lineární zrychlení. V některých případech může dojí ke zrychlení, které je větší něž ideální, tomuto zrychlení se říká superlineární zrychlení. To je nejčastěji zapříčiněno architekturou počítačů nebo rychlejším vyhledávaní hodnoty v menším souboru dat vlivem rozkladu oblasti (např. vyhledávání v seznamu uzlů sítě). Vliv architektury počítačů je hlavně dán velikostí paměti RAM, pokud při sekvenčním výpočtu dojde ke swapování na disk, doba výpočtu výrazně naroste. Dále také pamětí cache, při rozkladu oblasti se zmenšuje soubor dat, takže vetší část výpočtu probíhá za pomoci právě cache paměti, která je výrazně rychlejší než RAM. Nakonec ale zrychlení s rostoucím počtem procesorů začne klesat pod hodnotu ideálního zrychlení vlivem režie (komunikace mezi jádry, čekáni na výsledky ostatních jader, atd.). Z praktických zkušeností se doporučuje dodržet minimální velikost subdomény řádově 100 tisíc elementů, při větším počtu menších subdomén režie začne převažovat vlastní výpočet a zrychlení prudce klesá.

Dalším parametrem, který se určuje u paralelních výpočtů je efektivita. Efektivita je dána vztahem

$$E = \frac{T_S}{p * T_p} \tag{6.2}$$

a udává zrychlení na jedno jádro, tedy po jaký čas vykonává jádro (procesor) užitečnou práci, a jak velká část celkového výpočetního času připadá na režii a latenci. Hodnota efektivity se teoreticky pohybuje od 0 do 1, při hodnotě jedna je dosaženo ideálního zrychlení. Výjimkou je efektivita pro superlineární zrychlení, kde je E>1. [15]

Pro určení parametrů paralelního výpočtu byly zvoleny dvě úlohy proudění. *Úloha 1* je výpočet rychlostního a tlakového pole pomocí řešiče SIMPLE, délka řešení je 10 iterací a velikost sítě 3,25 milionu čtyřstěných elementů. *Úloha 2* se skládala ze stejného typu elementů a byl použit i stejný řešič, ale počet elementů byl výrazně menší, pouze 957 tisíc elementů. Délka řešení zde byla prodloužena na 30 iterací (viz tab. 6.2).

Tab. 6.2: Přehled řešených úloh

	počet elementů sítě	počet uzlů sítě	počet iterací	typ elementu	solver
úloha 1	3250016	531483	10	čtyřstěn	SIMPLE
úloha 2	957296	160639	30		

Před porovnáním zrychlení a efektivity je nejdřív nutné zjistit výpočetní čas dané úlohy při sekvenčním výpočtu. Tato hodnota je klíčová a odvíjí se od ní parametry paralelního výpočtu.

Nejprve se provedl postupně sekvenční výpočet na všech výpočetních uzlech, aby se stanovil průměrný čas sekvenčního výpočtu a zatížení paměti. Sledování paměti bylo spíše orientační, aby se ověřilo, že paměťové moduly o velikosti 4GB jsou dostačující a nedojde ke swapování na disk. Tím by byl výsledný čas sekvenčního výpočtu velice ovlivněn.

Průměrná doba sekvenčního výpočtu se pak určila pro oba typy procesorů zvlášť, jelikož uzly Intel XEON (viz tab. 6.3) dosahovaly lepšího času sekvenčního výpočtu (mají větší výpočetní výkon) než uzly AMD Opteron (viz tab. 6.4). Průměrné hodnoty výpočetních časů pro jednotlivé uzly jsou zaznamenány do tab. 6.5. Jak si můžeme všimnout u některých uzlů AMD Opteron (compute1-6, compute1-8 a compute1-9) je výpočetní čas výrazně odlišný, to by mělo za následek ovlivnění paralelních výpočtů, z tohoto důvodu se na uzlech compute1-6, compute1-8 a compute1-9 paralelní výpočet nespouštěl

Tab. 6.3: Čas sekvenčního výpočtu na uzlech Intel XEON

Intel XEON			
úloha 1		úloha 2	
uzel	doba výpočtu [s]	uzel	doba výpočtu [s]
compute0-0	554	compute0-0	385
compute0-1	556	compute0-1	393
compute0-2	551	compute0-2	385
compute0-3	551	compute0-3	385
compute0-4	553	compute0-4	384
compute0-5	554	compute0-5	386
compute0-6	554	compute0-6	385
compute0-7	553	compute0-7	383
compute0-8	563	compute0-8	385
compute0-9	551	compute0-9	385
compute0-10	554	compute0-10	387

Tab. 6.4: Čas sekvenčního výpočtu na uzlech AMD Opteron

AMD Opteron			
úloha 1		úloha 2	
uzel	doba výpočtu [s]	uzel	doba výpočtu [s]
compute1-0	716	compute1-0	548
compute1-1	704	compute1-1	530
compute1-2	693	compute1-2	531
compute1-4	707	compute1-4	542
compute1-5	693	compute1-5	537
compute1-6	767	compute1-6	587
compute1-7	707	compute1-7	541
compute1-8	760	compute1-8	596
compute1-9	780	compute1-9	583
compute1-10	702	compute1-10	532
compute1-12	707	compute1-12	537
compute1-13	700	compute1-13	539
compute1-14	699	compute1-14	529
compute1-15	698	compute1-15	534
compute1-16	692	compute1-16	530

Tab. 6.5: Průměrná doba sekvenčního výpočtu na jednotlivých uzlech

	průměrná doba sekvenčního výpočtu na uzlech Intel XEON	průměrná doba sekvenčního výpočtu na uzlech AMD Opteron
úloha 1	553 s	701 s
úloha 2	386 s	536 s

Jelikož je výpočetní cluster hybridní a pro rozklad oblasti se použily různé metody rozkladu oblasti (SIMPLE a METIS), je součásti této práce porovnání daných parametrů v závislosti jak na hardwaru tak na metodě rozkladu oblasti.

6.3 Testování paralelních výpočtů pomocí úlohy 1

Po provedení sekvenčního výpočtu následovaly paralelní výpočty pro úlohu 1. U paralelních výpočtů se opět sledoval a zaznamenával výpočetní čas a vytížení paměti. Paralelní výpočty se postupně spouštěly na výpočetních uzlech Intel XEON při vytížení všech jader v uzlu (tab. 6.6), to znamená, že na uzlu byla využita všechna čtyři jádra, jedná se o dva dvoujádrové procesory. Poté se vytížení uzlu snížilo na polovinu, byla využita dvě jádra v uzlu, na každém procesoru jedno jádro (tab. 6.7). Nakonec se pak využilo pouze jedno jádro v uzlu (tab. 6.8). Po skončení výpočtů na uzlech Intel XEON se výpočty spouštěly na uzlech AMD Opteron. Nejprve při plně vytížených uzlech AMD Opteron (tab. 6.9), dvě jádra na uzel, jedná se totiž o dva jednojádrové procesory. Poslední sada výpočtů pak proběhla při obsazení jednoho jádra v uzlu (tab. 6.10). Všechny testy se pak prováděly pro dvě metody rozkladu oblasti a to metodu *SIMPLE* a *METIS*, aby se mohlo porovnat, která z daných metod je pro dekompozici oblasti lepší.

Tab. 6.6: Výpočetní čas a vyžití paměti pro uzly Intel XEON, vytížení všech jader

INTEL XEON výpočet na všech jádrech v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	výpočetní čas [s]	využití paměti [MB]	výpočetní čas [s]
2	1089-1354	299	1065-1374	299
4	604-705	222	560-684	228
6	400-510	143	384-456	138
8	294-364	105	301-372	97
12	209-264	69	203-256	62
16	162-196	54	150-200	49
20	124-182	49	128-175	41
24	124-145	41	119-142	36
28	95-132	43	91-125	32
32	85-120	35	76-110	28
36	70-110	48	65-97	33

Tab. 6.7: Výpočetní čas a využití paměti pro uzly Intel XEON, vytížení dvou jader

INTEL XEON výpočet na dvou jádrech v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	výpočetní čas [s]	využití paměti [MB]	výpočetní čas [s]
2	1100-1386	300	1090-1350	299
4	590-680	132	570-650	136
6	380-490	87	400-480	85
8	310-390	70	306-365	63
10	250-310	63	224-297	50
12	217-265	51	195-250	44
14	194-240	51	183-244	37
16	160-212	48	157-204	32
18	154-194	53	142-183	32

Tab. 6.8: Výpočetní čas a využití paměti pro uzly Intel XEON, vytíženo jednoho jádra

INTEL XEON výpočet na jednom jádru v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	výpočetní čas [s]	využití paměti [MB]	výpočetní čas [s]
2	1230-1350	264	1150-1450	260
4	580-736	123	570-690	121
6	376-426	85	404-503	77
8	316-386	68	220-316	55

Tab. 6.9: Výpočetní čas a využití paměti pro uzly AMD Opteron, vytížení všech jader

AMD Opteron výpočet na všech jádrech v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	výpočetní čas [s]	využití paměti [MB]	výpočetní čas [s]
2	1080-1220	386	1100-1332	373
4	574-688	177	562-677	191
6	403-465	139	407-480	117
8	302-364	108	290-379	87
12	216-248	78	185-254	60
16	182-214	67	155-206	49
20	131-169	55	121-169	39
24	112-146	47	106-142	40
28	96-130	42	92-120	33

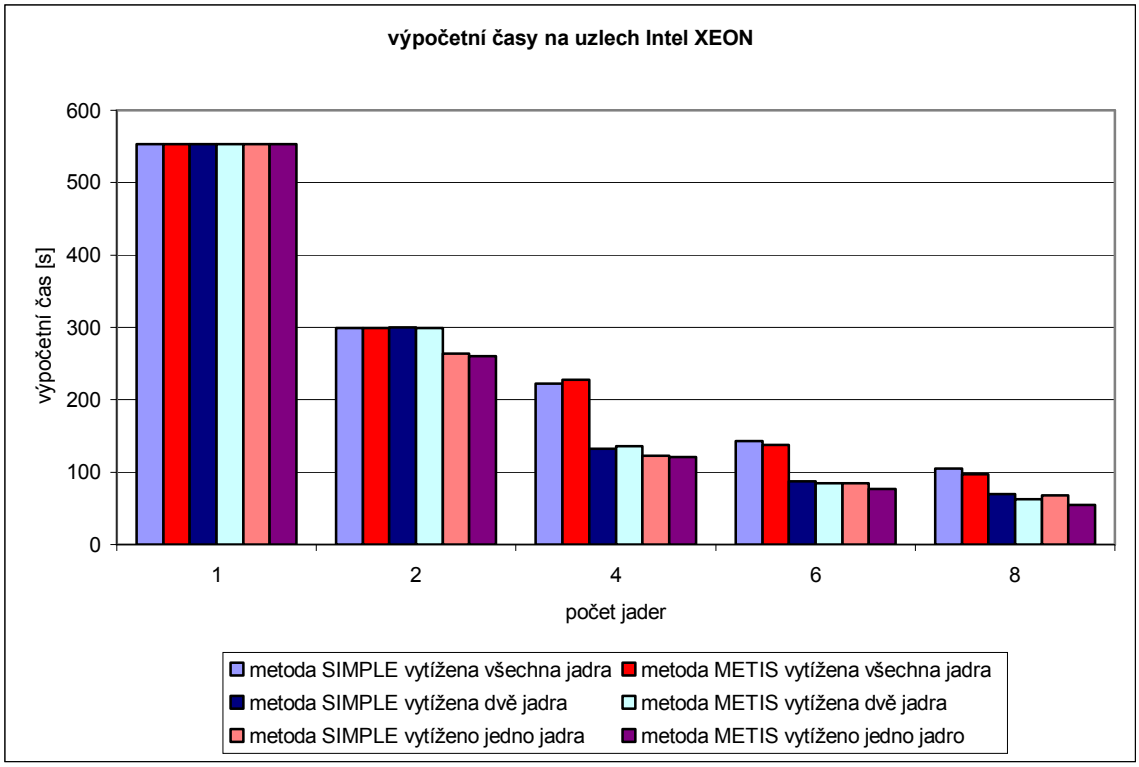
Tab. 6.10: Výpočetní čas a využití paměti pro uzly AMD Opteron, vytíženo jednoho jádra

AMD Opteron výpočet na jednom jádru v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	výpočetní čas [s]	využití paměti [MB]	výpočetní čas [s]
2	1090-1370	365	1090-1250	356
4	574-688	176	590-770	179
6	417-486	132	390-488	124
8	294-382	107	304-375	86
12	216-260	74	196-258	58

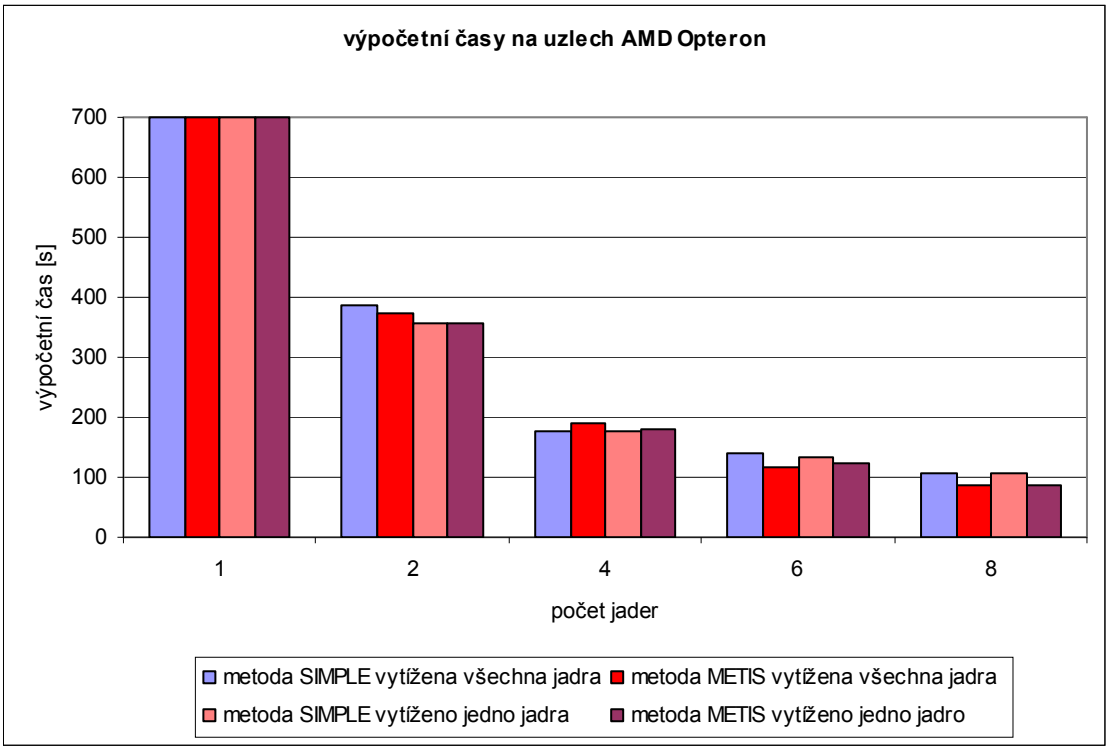
Pro lepší přehlednost a porovnání jsou hodnoty výpočetního času z tabulek vyneseny do grafů pro uzly Intel XEON (viz graf 6.1) a AMD Opteron (viz graf 6.2). V těchto grafech se zobrazuje výpočetní čas úlohy 1 pro 1 až 8 jader v závislosti na použité metodě dekompozice a vytížení uzlu.

Nejprve zhodnotíme vliv vytížení uzlu na výpočetní čas. V grafu 6.1 je jasně patrné, že u uzlů Intel XEON má vytížení uzlu výrazný vliv na výpočetní čas. Při výpočtu na osmi jádrech je výpočetní čas na plně obsazených uzlech téměř dvakrát delší, než když pro výpočet využijeme pouze jedno jádro v uzlu. Naproti tomu u uzlů AMD Opteron se vytížení uzlů na hodnotu výpočetní čas takřka neprojeví a výpočetní časy jsou téměř stejné.

Dále se pak hodnotil vliv metody dekompozice na výpočetní čas. Vliv metody dekompozice se začíná projevovat, až při větším počtu jader a s rostoucím počtem jader se rozdíl zvětšuje, protože se zvětšuje režie. Z grafů 6.1a 6.2 je patrné, že je lepší pro dekompozici oblast využít metodu METIS, dosáhneme kratšího výpočetního času než u metody SIMPLE.



Graf 6.1: Porovnání výpočetního času pro jednotlivé metody dekompozice a obsazení jader v uzlech Intel XEON



Graf 6.2: Porovnání výpočetního času pro jednotlivé metody dekompozice a obsazení jader v uzlech AMD Opteron

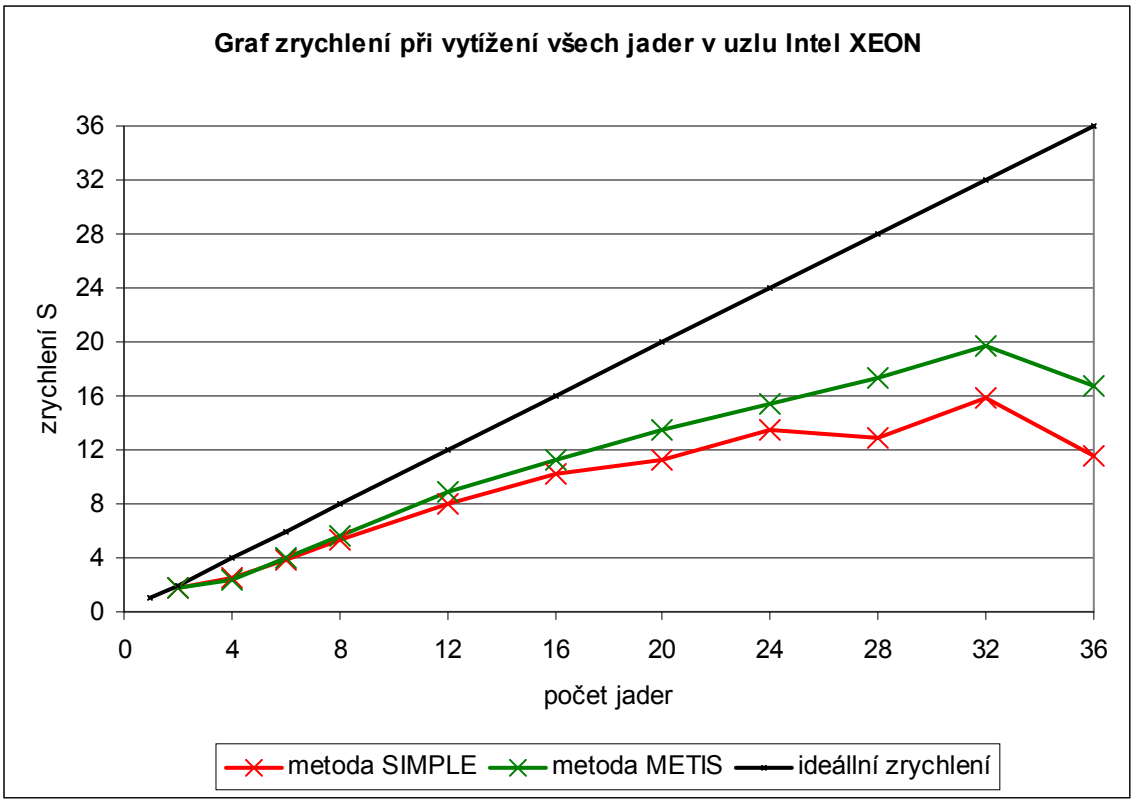
Po změření a zaznamenání jednotlivých výpočetních časů paralelních výpočtů do tabulek se začaly vyhodnocovat parametry paralelního výpočtu. Prvním parametrem, který se hodnotil, je zrychlení. Na hodnotu tohoto parametru má vliv jak metoda dekompozice oblasti, tak i vytížení uzlů, vše bylo shrnuto do tab. 6.11 pro uzly Intel XEON a do tab. 6.12 pro uzly AMD Opteron.

Tab. 6.11: Zrychlení pro jednotlivé metody dekompozice a vytížení uzlů, Intel XEON

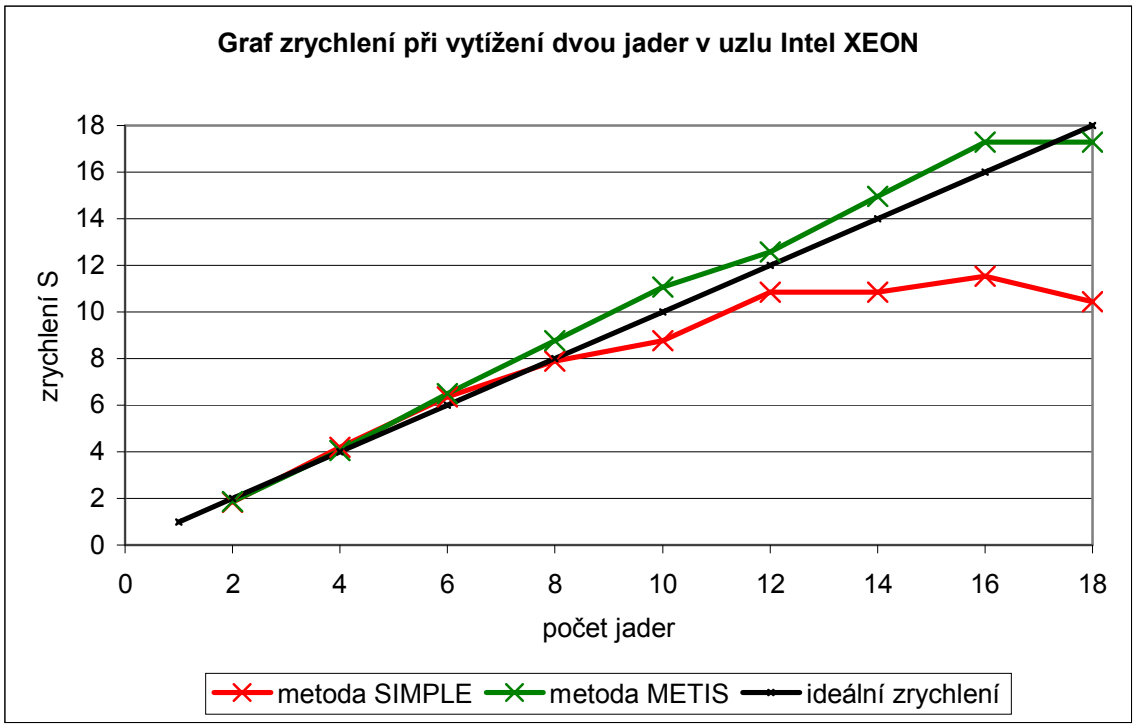
počet jader	zrychlení výpočtu na procesorech Intel XEON					
	zatížena všechna jádra v uzlu		zatížena dvě jádra v uzlu		zatíženo jedno jádro v uzlu	
	METIS	SIMPLE	METIS	SIMPLE	METIS	SIMPLE
2	1,85	1,85	1,85	1,84	2,13	2,09
4	2,43	2,49	4,07	4,19	4,57	4,50
6	4,01	3,87	6,51	6,36	7,18	6,51
8	5,70	5,27	8,78	7,90	10,05	8,13
12	8,92	8,01	12,57	10,84	-	-
16	11,29	10,24	17,28	11,52	-	-
20	13,49	11,29	-	-	-	-
24	15,36	13,49	-	-	-	-
28	17,28	12,86	-	-	-	-
32	19,75	15,80	-	-	-	-
36	16,76	11,52	-	-	-	-

Tab. 6.12: Zrychlení pro jednotlivé metody dekompozice a vytížení uzlů, AMD Opteron

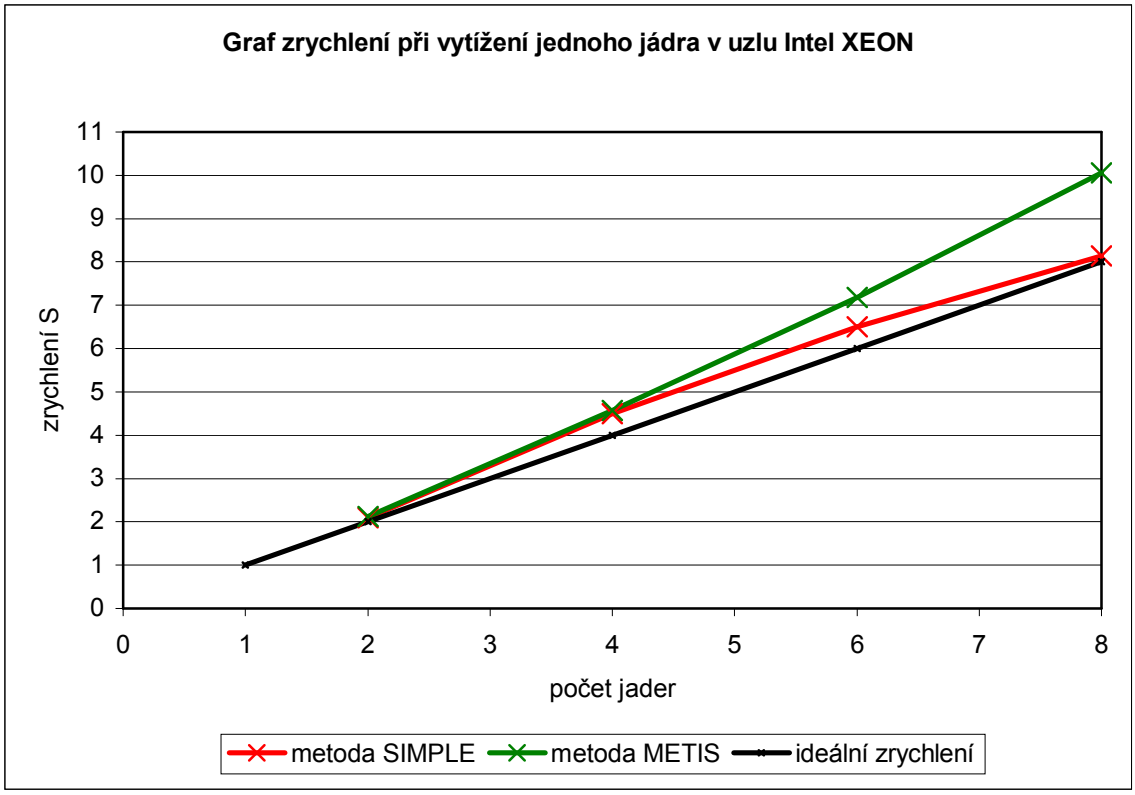
počet jader	zrychlení výpočtu na procesorech AMD Opteron			
	zatížena všechna jádra v uzlu		zatíženo jedno jádra v uzlu	
	METIS	SIMPLE	METIS	SIMPLE
2	1,88	1,82	1,97	1,92
4	3,67	3,96	3,92	3,98
6	5,99	5,04	5,65	5,31
8	8,06	6,49	8,15	6,55
12	11,68	8,99	12,09	9,47
16	14,31	10,46	-	-
20	17,97	12,75	-	-
24	17,53	14,91	-	-
28	21,24	16,69	-	-



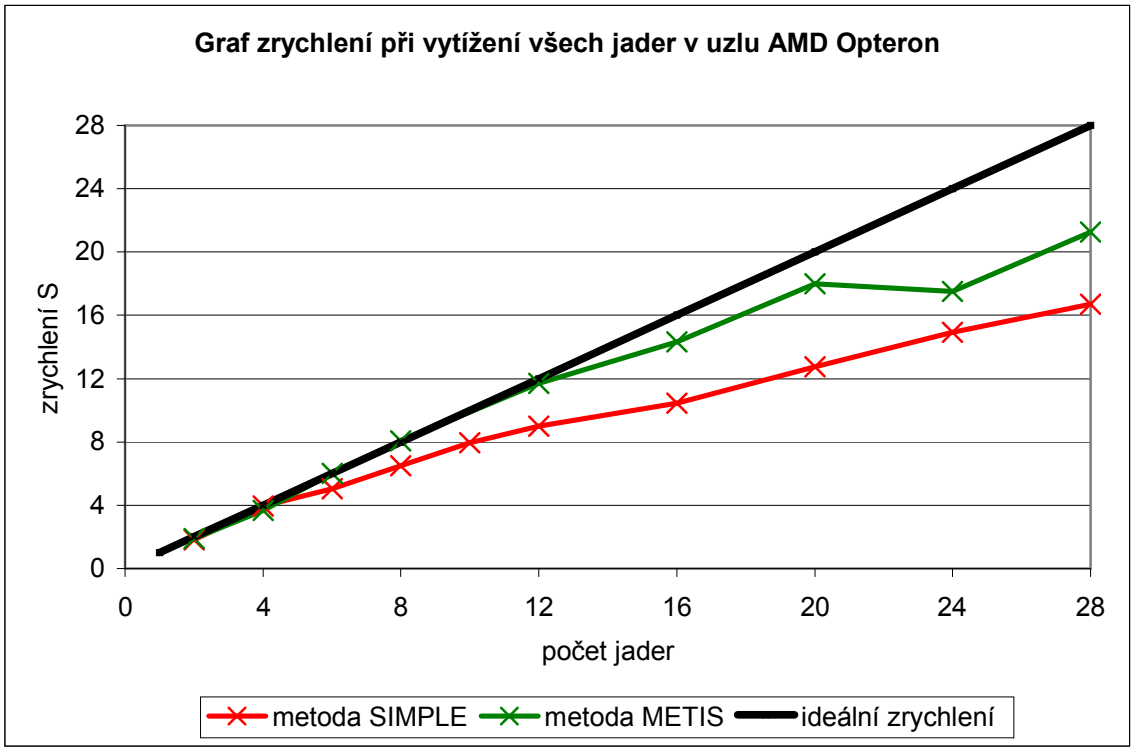
Graf 6.3: Porovnání zrychlení pro jednotlivé metody dekompozice při vytížení všech jader v uzlech Intel XEON



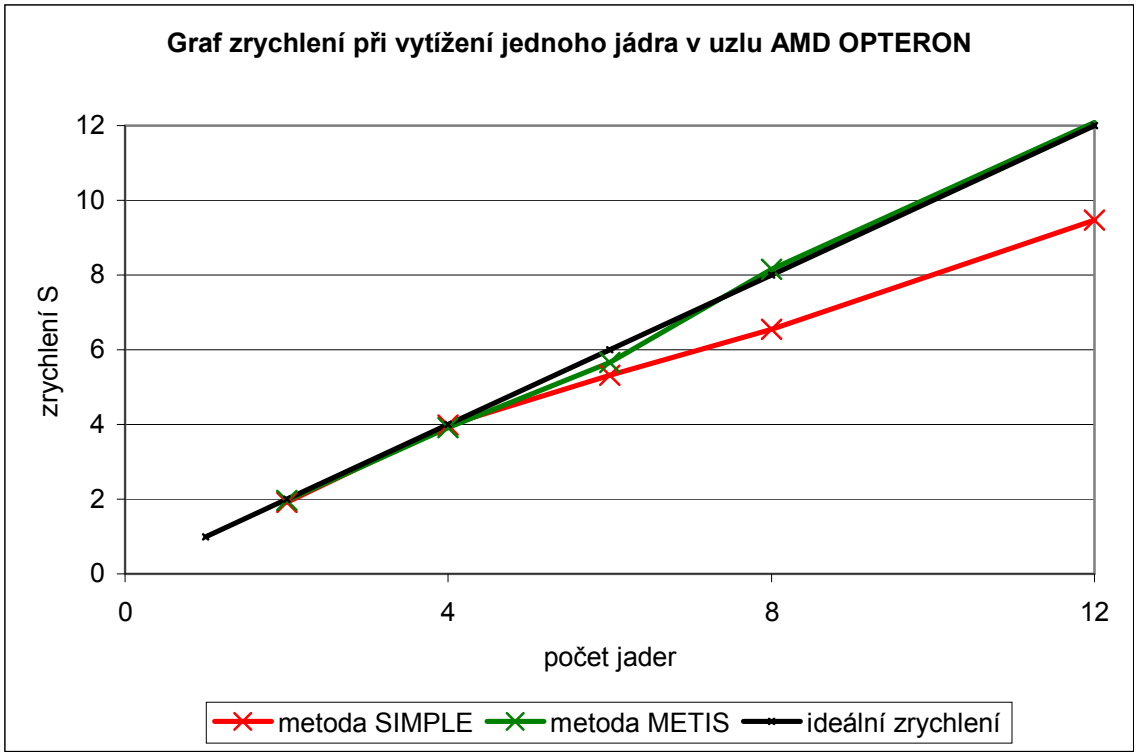
Graf 6.4: Porovnání zrychlení pro jednotlivé metody dekompozice při vytížení dvou jader v uzlech Intel XEON



Graf 6.5: Porovnání zrychlení pro jednotlivé metody dekompozice při vytížení jednoho jádra v uzlech Intel XEON



Graf 6.6: Porovnání zrychlení pro jednotlivé metody dekompozice při vytížení všech jader v uzlech AMD Opteron



Graf 6.7: Porovnání zrychlení pro jednotlivé metody dekompozice při vytížení jednoho jádra v uzlech AMD Opteron

Vliv dekompozice oblasti na zrychlení je patrný u všech grafů (viz graf 6.3 až 6.7). Jak je vidět, metoda dekompozice oblasti METIS je pro rozklad oblasti lepší (zelená čára v grafech), dosahuje se při ní vyššího zrychlení než u metody SIMPLE (červená čára v grafech). S rostoucím počtem jader je rozdíl mezi oběma metodami výraznější, to je způsobeno tím, že náklady na režii jsou v případě metody METIS menší než u metody SIMPLE.

Vliv vytížení uzlů na zrychlení je dán typem výpočetního uzlu. Výpočetní uzly se od sebe odlišují. To má za následek, že u uzlů AMD Opteron je vliv vytížení uzlu na zrychlení zanedbatelný, zatím co u uzlů Intel XEON je naopak teto vliv dominantní.

Při vytížení všech jader v uzlu Intel XEON (viz graf 6.3), vidíme, že je zrychlení menší než ideální, s rostoucím počtem jader se tento rozdíl zvětšuje a zrychlení roste jen velmi pomalu. Tento způsob vytížení uzlu nám poskytuje nejmenší zrychlení. Pokud se, ale k výpočtu využije v obou procesorech jen jedno jádro (viz graf 6.4), je pro metodu dekompozice METIS dosaženo zrychlení superlinéarní, zrychlení je tedy vyšší než ideální. Nejlepšího zrychlení paralelního výpočtu docílíme, pokud k vypočtu využijeme jen jedno jádro a jeden procesor v uzlu (viz graf 6.5). Zrychlení je výrazně větší než ideální, pro výpočet na 8 jádrech a použití metody dekompozice METIS se výpočet

zrychlí více než 10-krát. To je nejspíše způsobeno tím, že uzly Intel XEON mají sdílenou paměť L2-cache a pokud máme k výpočtu použita obě jádra v procesoru, musí obě jádra využít více paměť RAM, která je oproti paměti cache pomalejší. Další vliv na zrychlení má také samotná paměť RAM, protože uzel Intel XEON využívá pro oba procesory jednu sdílená paměť. To má za následek, že s paměti může komunikovat v daném čase pouze jedno jádro a ostatní pak musí čekat a využít svojí paměť cache.

U uzlů AMD Opteron je vliv vytížení jader na zrychlení zanedbatelný. Pro stejný počet jader se získá přibližně stejné zrychlení jak pro plně vytížený uzel, vypočet je spuštěn na obou procesorech v uzlu (viz graf 6.6), tak i při vytížení jednoho jádra v uzlu, výpočet je spouštěn na jednom procesoru v uzlu (viz graf 6.7). Architektura těchto procesorů je totiž oproti Intel XEON rozdílná v tom, že přístup do paměti je NUMA (Non-Uniform Memory Access), tedy každý procesor má vlastní přístup do paměti RAM, tím pádem se navzájem neomezují. Navíc mají uzly AMD nativně implementované hardwarové linky pro meziprocesorovou komunikaci. Výsledkem je, že je skoro lepší spustit výpočet na míň plně vytížených uzlech. Z tohoto pohledu mají uzly AMD Opteron oproti uzlům Intel XEON výhodu, jelikož využijeme plně potenciál výpočetního zařízení bez omezení a tím pádem je využíváno mnohem hospodárněji.

Po vyhodnocení všech výsledků zrychlení pro úlohu 1 se jeví jako nejlepší kombinace hardwaru a metody dekompozice spouštět paralelní výpočet, při použití metody dekompozice METIS a využít plně obsazené uzly AMD Opteron nebo uzly Intel XEON a výpočty zde spouštět na jednom jádru pro každý procesor.

Druhým parametrem, který se určuje u paralelních výpočtů, je efektivita, opět byl porovnáván vliv jak metody dekompozice, tak i vytížení uzlů. Hodnoty efektivity byly zapsány zvlášť pro uzly Intel XEON (viz tab. 6.13) a pro uzly AMD Opteron (viz tab. 6.14).

Tab. 6.13: Efektivita pro jednotlivé metody dekompozice a vytížení uzlů pro uzly Intel XEON

Počet jader	Efektivita výpočtu na procesorech Intel XEON					
	zatížena všechna jádra v uzlu		zatížena všechna jádra v uzlu		zatíženo jedno jádra v uzlu	
	METIS	SIMPLE	METIS	SIMPLE	METIS	SIMPLE
2	0,92	0,92	0,92	0,92	1,06	1,05
4	0,61	0,62	1,02	1,05	1,14	1,12
6	0,67	0,64	1,08	1,06	1,20	1,08
8	0,71	0,66	1,10	0,99	1,26	1,02
12	0,74	0,67	1,05	0,90	-	-
16	0,71	0,64	1,08	0,72	-	-
20	0,67	0,56	-	-	-	-
24	0,64	0,56	-	-	-	-
28	0,62	0,46	-	-	-	-
32	0,62	0,49	-	-	-	-
36	0,47	0,32	-	-	-	-

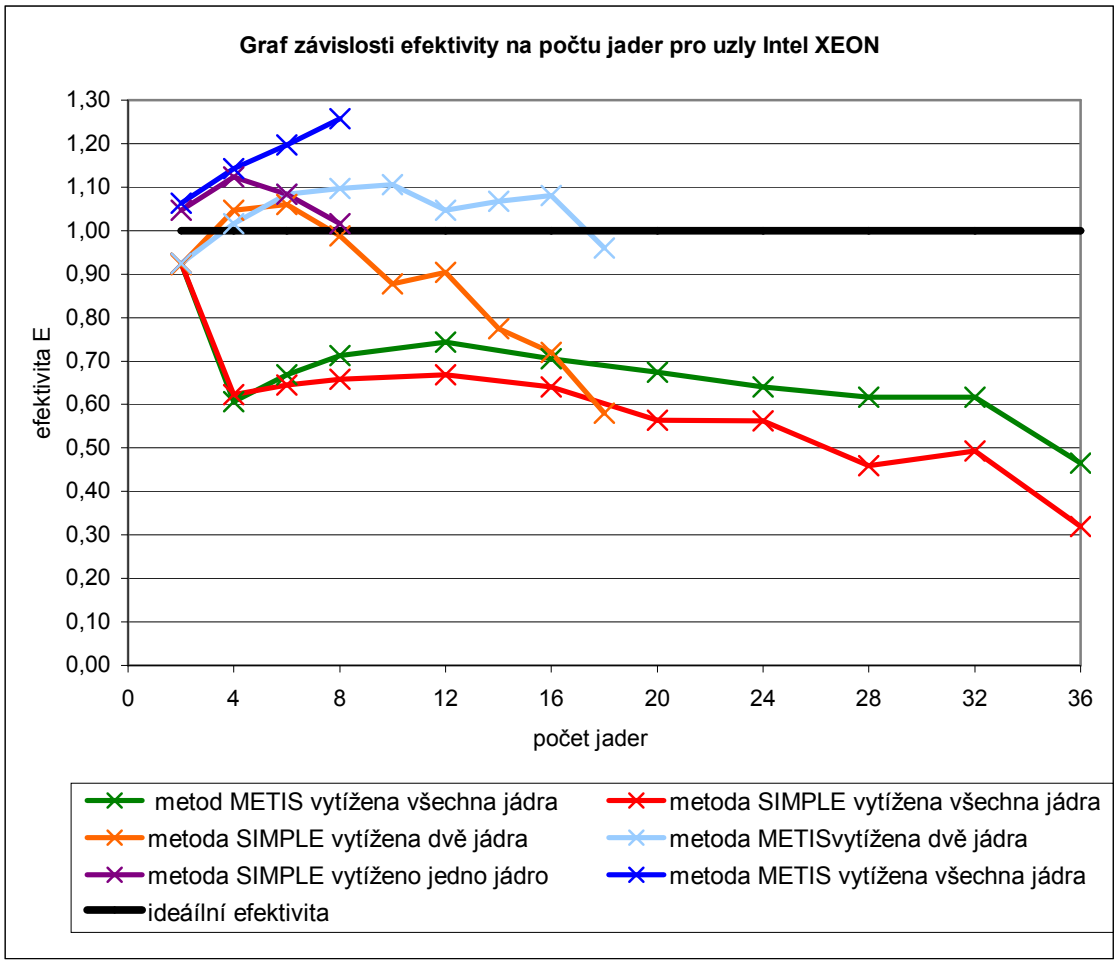
Tab. 6.14: Efektivity pro jednotlivé metody dekompozice a vytížení uzlů pro uzly AMD Opteron

Počet jader	Efektivita výpočtu na procesorech AMD OPTERON			
	zatížena všechna jádra v uzlu		zatíženo jedno jádra v uzlu	
	METIS	SIMPLE	METIS	SIMPLE
2	0,94	0,91	0,98	0,96
4	0,92	0,99	0,98	1,00
6	1,00	0,84	0,94	0,89
8	1,01	0,81	1,02	0,82
12	0,97	0,75	1,01	0,79
16	0,89	0,65	-	-
20	0,90	0,64	-	-
24	0,73	0,62	-	-
28	0,76	0,60	-	-

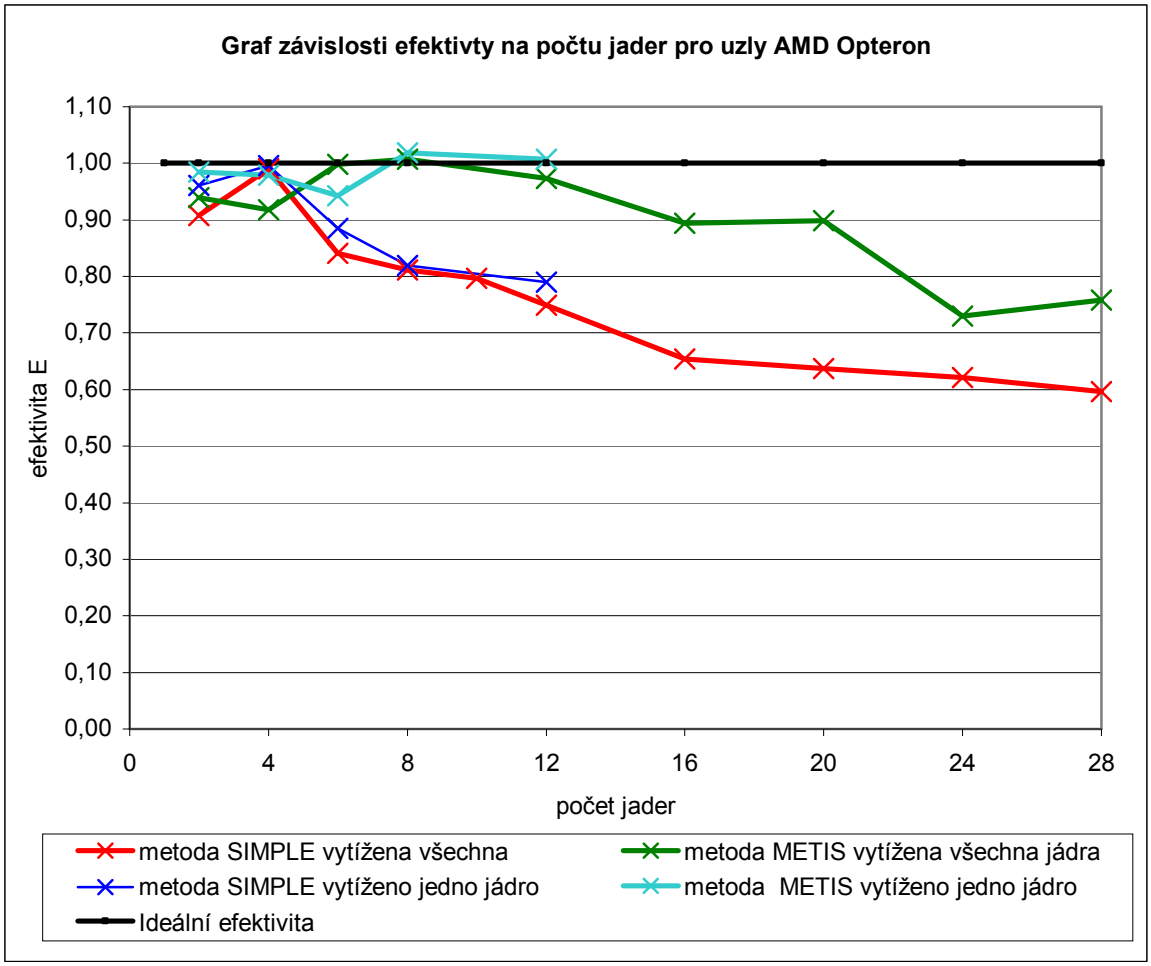
Následně se získané hodnoty vynesly do grafu pro každý typ uzlů jednotlivě. Z grafu 6.8, kde je vynesena závislost efektivity na počtu jader pro uzly Intel XEON, je názorně vidět, že největší efektivity se dosahuje při použití metody dekompozice METIS a při vytížení jednoho jádra v uzlu (tmavě modrá barva). Naopak při vytížení všech jader a použití metody dekompozice SIMPLE je efektivita velmi malá a jádro je využito k výpočtu pouze 1/3 z celkového času a zbytek pak připadá na režii (červená barva v grafu). Proto počítat tuto úlohu je, při takto zvolené metodě dekompozice a hardwaru, na více jak 24 jádrech neekonomické. V případě metody dekompozice METIS a plného vytížení uzlu je ekonomické počítat maximálně na 32 jádrech. Zde

vidím hlavní přínos metody METIS oproti metodě SIMPLE. Celkově se za pomoci metody dekompozice METIS dosahuje lepších výsledků, efektivita neklesá s rostoucím počtem jader tak rychle, jako je tomu v případě metody SIMPLE.

Výhodou uzlů AMD Opteron je, že při plně vytížených uzlech (využijeme plně kapacitu clusteru) se dosahuje vysoké efektivity i při velkém počtu jader (viz graf 6.9). U metody dekompozice METIS se dosahuje vysoké efektivity, i při výpočtu na 28 jádrech je efektivita 0,75.



Graf 6.8: Porovnání efektivity pro jednotlivé metody dekompozice a různém vytížení uzlů na uzlech Intel XEON



Graf 6.9: Porovnání efektivity pro jednotlivé metody dekompozice a různém vytížení uzlů na uzlech AMD Opteron

6.4 Testování paralelních výpočtů pomocí úlohy 2

Pro ověření výsledků se provedlo stejné testování i pro druhou úlohu, která měla oproti úloze 1 výrazně menší počet elementů (viz tab. 6.2). Provedlo se tedy testovaní vlivu metody dekompozice oblasti a vytížení uzlů na výpočetní čas, zrychlení a efektivitu paralelního výpočtu dané úlohy. Hodnoty výpočetního času a využití paměti byly zaznamenány do tabulek, nejprve pro výpočty na uzlech Intel XEON při vytížení všech jader v uzlu (viz tab. 6.15), při vytížení dvou jader v uzlu (viz tab. 6.16) a nakonec i při vytížení jednoho jádra v uzlu (tab. 6.17). Poté se provedly i výpočty na uzlech AMD Opteron a zaznamenaly se hodnoty výpočetního času a využití paměti nejprve při vytížení všech jader v uzlech (viz tab. 6.18) a také při vytížení jednoho jádra v uzlu (viz tab. 6.19).

Tab. 6.15: Výpočetní čas a využití paměti pro uzly Intel XEON, vytížení všech jader

INTEL XEON výpočet na všech jádrech v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	doba výpočtu [s]	využití paměti [MB]	doba výpočtu [s]
2	365-420	192	356-434	190
4	175-245	139	185-208	140
6	126-165	88	120-143	84
8	110-122	64	105-124	60
12	65-90	42	71-84	45
16	56-72	43	50-65	36
20	42-62	33	42-54	31
24	40-44	34	35-62	31
28	31-42	33	30-43	34
32	28-40	36	25-35	26
36	26-36	33	23-32	28

Tab. 6.16: Výpočetní čas a využití paměti pro uzly Intel XEON, vytížení dvou jader

INTEL XEON výpočet na dvou jádrech v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	doba výpočtu [s]	využití paměti [MB]	doba výpočtu [s]
2	356-433	192	345-397	190
4	182-224	84	185-225	83
6	142-167	56	124-147	53
8	109-119	45	96-124	43
10	82-102	40	74-100	34
12	76-85	35	62-88	29
14	64-79	32	63-75	27
16	52-68	37	52-70	24
18	46-62	33	46-62	24

Tab. 6.17: Výpočetní čas a využití paměti pro uzly Intel XEON, vytíženo jednoho jádra

INTEL XEON výpočet na jednom jádru v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	doba výpočtu [s]	využití paměti [MB]	doba výpočtu [s]
2	356-405	171	361-420	170
4	184-224	83	183-212	78
6	124-158	57	127-161	50
8	100-125	47	99-117	37

Tab. 6.18: Výpočetní čas a využití paměti pro uzly AMD Opteron, vytížení všech jader

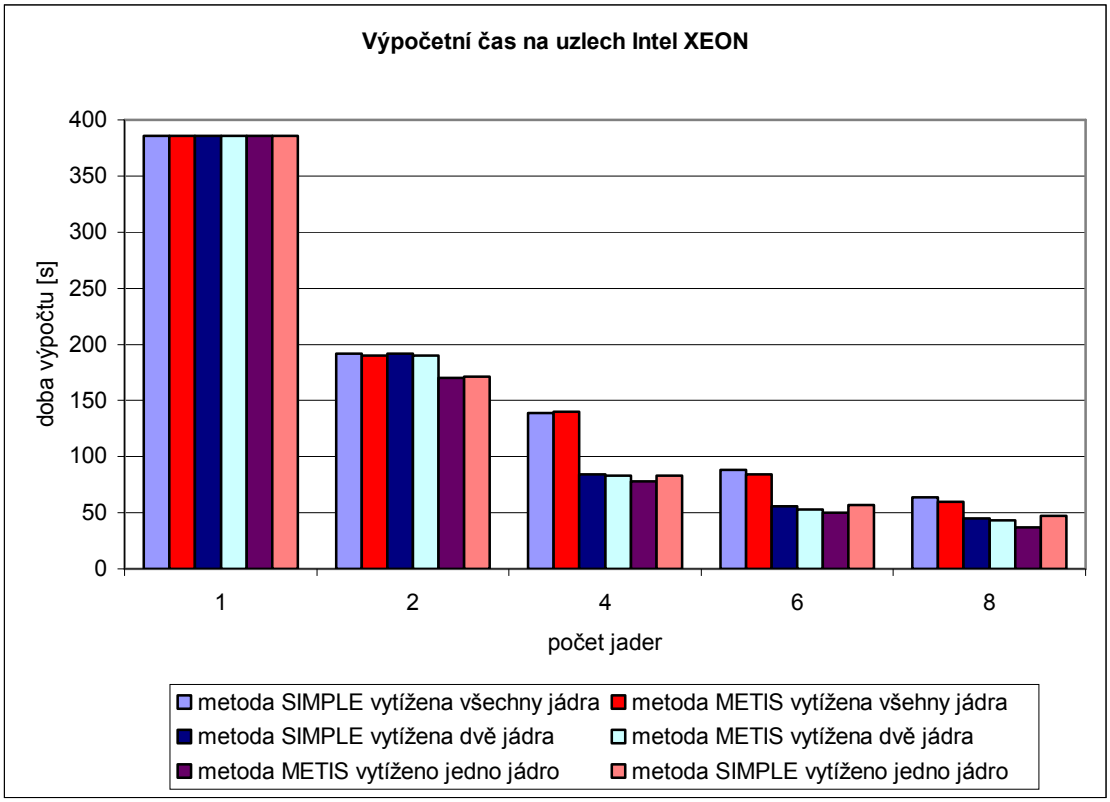
AMD Opteron výpočet na všech jádrech v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	doba výpočtu [s]	využití paměti [MB]	doba výpočtu [s]
2	356-389	273	351-432	272
4	176-224	138	180-240	138
6	127-148	88	115-164	91
8	109-127	72	94-126	74
12	69-98	59	63-85	53
16	52-66	58	49-65	36
20	42-57	40	43-62	32
24	39-46	43	36-45	30
28	32-40	58	23-43	30

Tab. 6.19: Výpočetní čas a využití paměti pro uzly AMD Opteron, vytíženo jednoho jádra

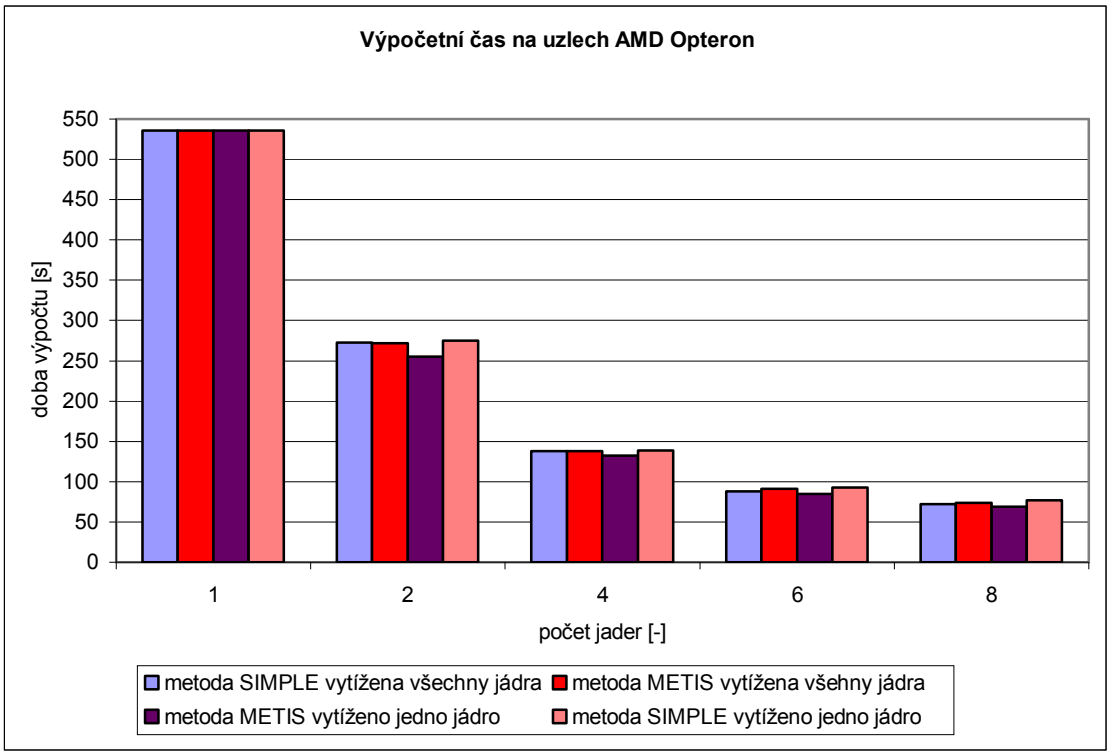
AMD Opteron výpočet na jednom jádru v uzlu				
počet jader	metoda dekompozice SIMPLE		metoda dekompozice METIS	
	využití paměti [MB]	doba výpočtu [s]	využití paměti [MB]	doba výpočtu [s]
2	335-456	275	359-412	255
4	171-226	139	174-237	132
6	142-167	93	103-158	85
8	105-124	77	89-124	69
12	78-105	61	78-105	45

Stejně jako u předešlé úlohy se naměřené hodnoty výpočetního času vynesly do grafu. Z grafů (viz graf 6.10 a 6.11) je vidět, že jsme získali přibližně stejné výsledky jako v úloze 1. Metoda dekompozice METIS tedy poskytuje lepší výsledky (kratší výpočetní čas) než metoda SIMPLE, i když rozdíl mezi metodami je zde menší než v případě úlohy 1.

Přibližně stejné výsledky se projevily i u vlivu vytížení uzlů. U výpočetního času se pro uzly Intel XEON (viz graf 6.10) dosáhlo nejkratšího výpočetního času pro daný počet jader při vytížení jednoho jádra v uzlu a metodě dekompizice METIS a nejdelšího výpočetního času při plně vytíženém uzlu a metodě dekompozice SIMPLE. Pro uzly AMD Opteron (viz grafu 6.11), se opět vliv obsazení jádra takřka neprojevil.



Graf 6.10: Porovnání výpočetního času úlohy 2 pro jednotlivé metody dekompozice a vytížení jader v uzlech Intel XEON



Graf 6.11: Porovnání výpočetního času úlohy 2 pro jednotlivé metody dekompozice a vytížení jader v uzlech AMD Opteron

Po ověření vlivu metody dekompozice a vytížení uzlu na výpočetní čas je jasné, že stejných výsledků jako v případě úlohy 1 docílíme i u zrychlení, protože zrychlení je dáno výpočetním časem (6.1). Hodnoty zrychlení se zaznamenaly zvlášť pro uzly Intel XEON (viz tab. 6.20) a AMD Opteron (viz tab. 6.21).

Nejvyššího zrychlení se pro uzly Intel XEON opět dosáhlo při použití metody dekompozice METIS a při vytížení jednoho jádra v uzlu (viz graf 6.14). Zrychlení bylo opět superlineálrní a při výpočtu na 8 jádrech bylo dosaženo zrychlení 10,43. Nejmenšího zrychlení bylo dosaženo pro metodu dekompozice SIMPLE při plně vytíženém uzlu (viz graf 6.12), ale rozdíl mezi metodami dekompozice byl v tomto případ výrazně menší než tomu bylo u úlohy 1.

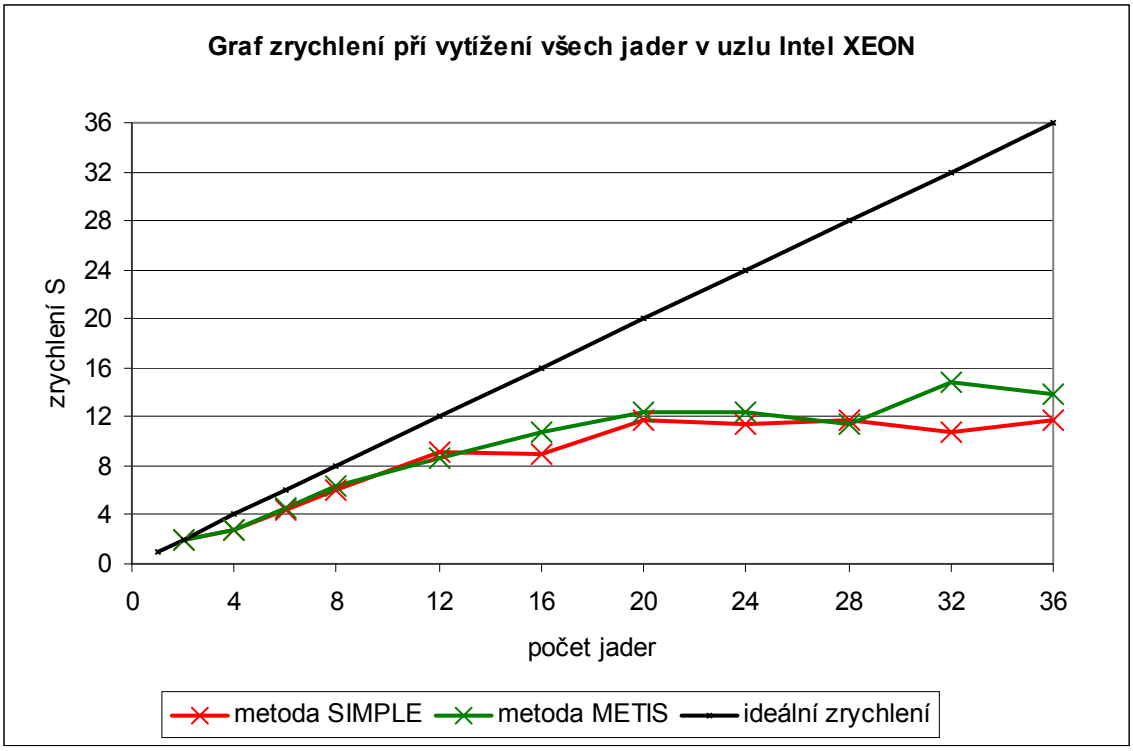
Podobné výsledky pro zrychlení se v porovnání s úlohou 1 dosáhly i na uzlech AMD Opteron. Vliv vytížení uzlu byl opět zanedbatelný a lepší výsledky byly získány při použití metody dekompozice METIS. Rozdíl v porovnání s předchozí *úlohou 1* je ten, že při použití metody dekompozice SIMPLE, začalo zrychlení dokonce klesat (viz graf 6.15), jelikož velikost podoblasti byla tak malá, že přínos zrychlení výpočtu byl menší než náklady na režii. Pro rozklad na 20 subdomén, jedna subdoména obsahovala 48 tisíc elementů.

Tab. 6.20: Zrychlení pro jednotlivé metody dekompozice a vytížení uzlů, Intel XEON

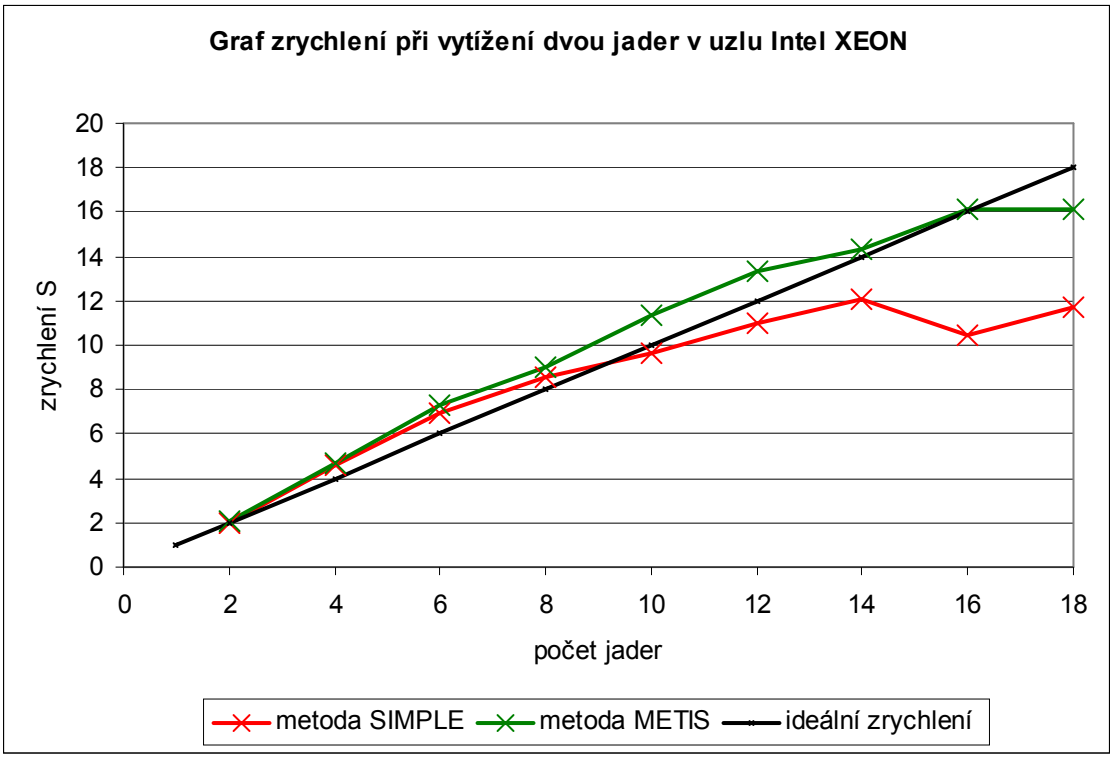
počet jader	zrychlení výpočtu na procesorech Intel XEON					
	zatížena všechna jádra v uzlu		zatížena dvě jádra v uzlu		zatíženo jedno jádra v uzlu	
	METIS	SIMPLE	METIS	SIMPLE	METIS	SIMPLE
2	2,03	2,01	1,85	2,01	2,27	2,26
4	2,76	2,78	4,07	4,60	4,95	4,65
6	4,60	4,39	6,51	6,89	7,72	6,77
8	6,43	6,03	8,78	8,58	10,43	8,21
12	8,58	9,19	12,57	11,03	-	-
16	10,72	8,98	16,08	10,43	-	-
20	12,45	11,70	-	-	-	-
24	12,45	11,35	-	-	-	-
28	11,35	11,70	-	-	-	-
32	14,85	10,72	-	-	-	-
36	13,79	11,70	-	-	-	-

Tab. 6.21: Zrychlení pro jednotlivé metody dekompozice a vytížení uzlů , AMD Opteron

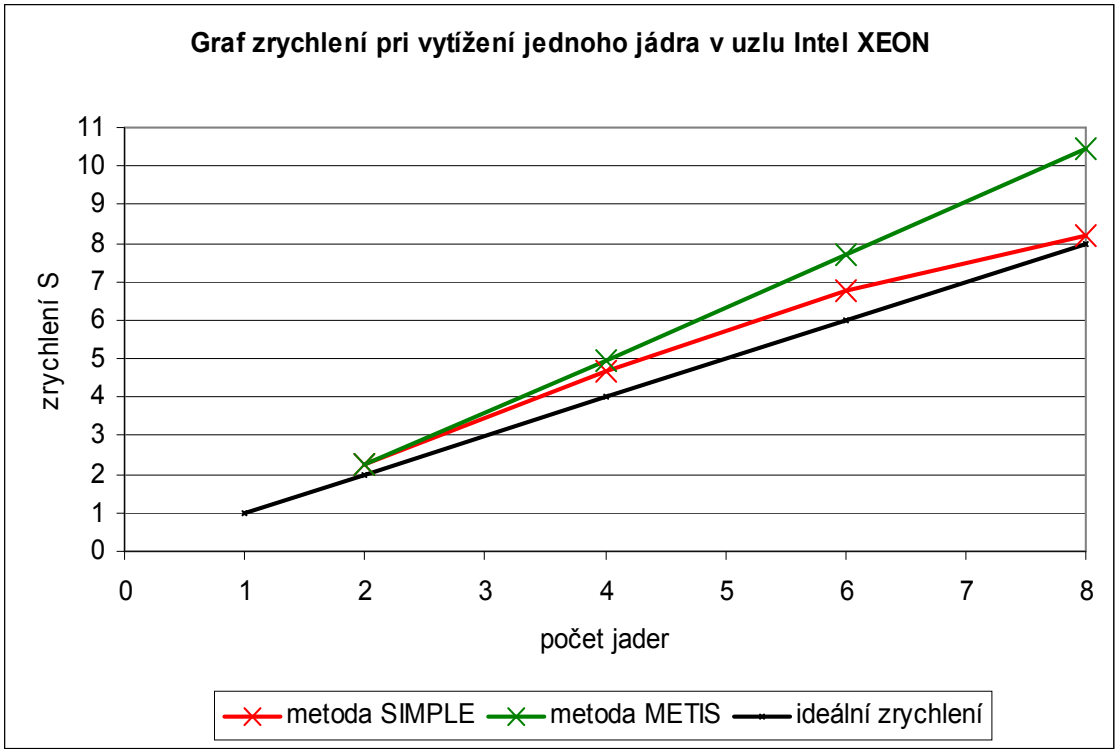
počet jader	zrychlení výpočtu na procesorech AMD Opteron			
	zatížena všechna jádra v uzlu		zatíženo jedno jádra v uzlu	
	METIS	SIMPLE	METIS	SIMPLE
2	1,97	1,96	2,10	1,95
4	3,88	3,88	4,06	3,86
6	5,89	6,09	6,31	5,76
8	7,24	7,44	7,77	6,96
12	10,11	9,08	11,91	9,24
16	14,89	9,24	-	-
20	16,75	13,40	-	-
24	17,87	12,47	-	-
28	17,87	9,24	-	-



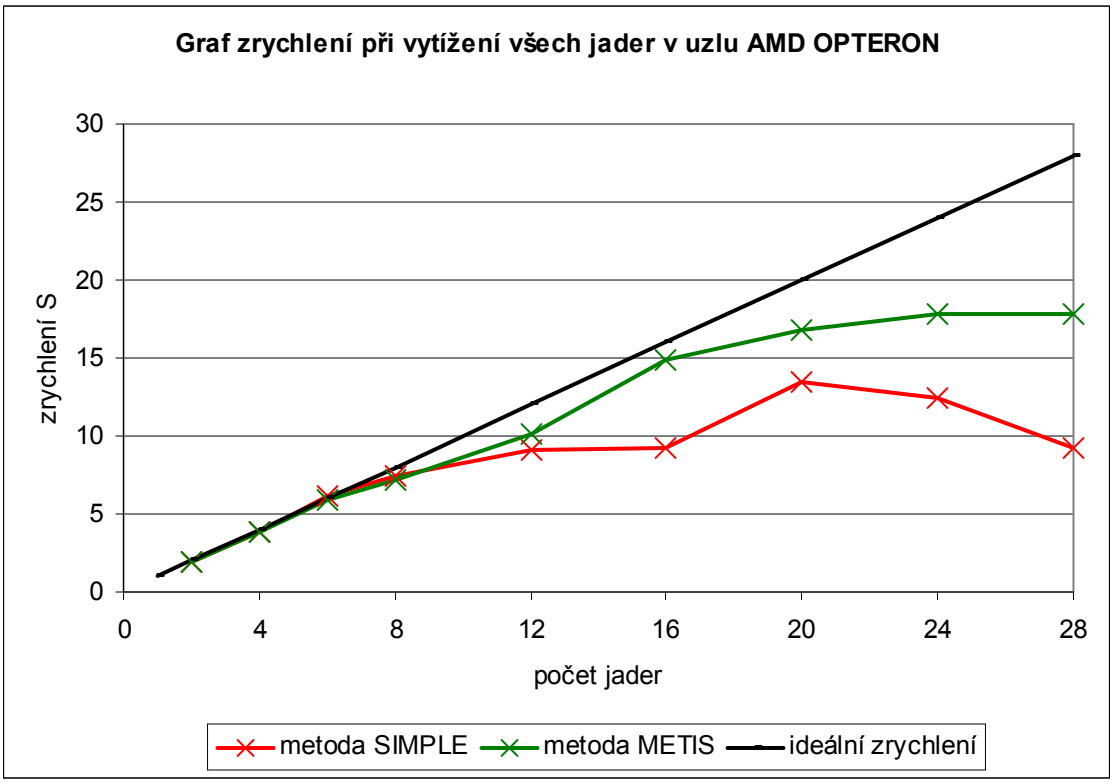
Graf 6.12: Zrychlení pro jednotlivé metody dekompozice při vytížení všech jader v uzlu na uzlech Intel XEON



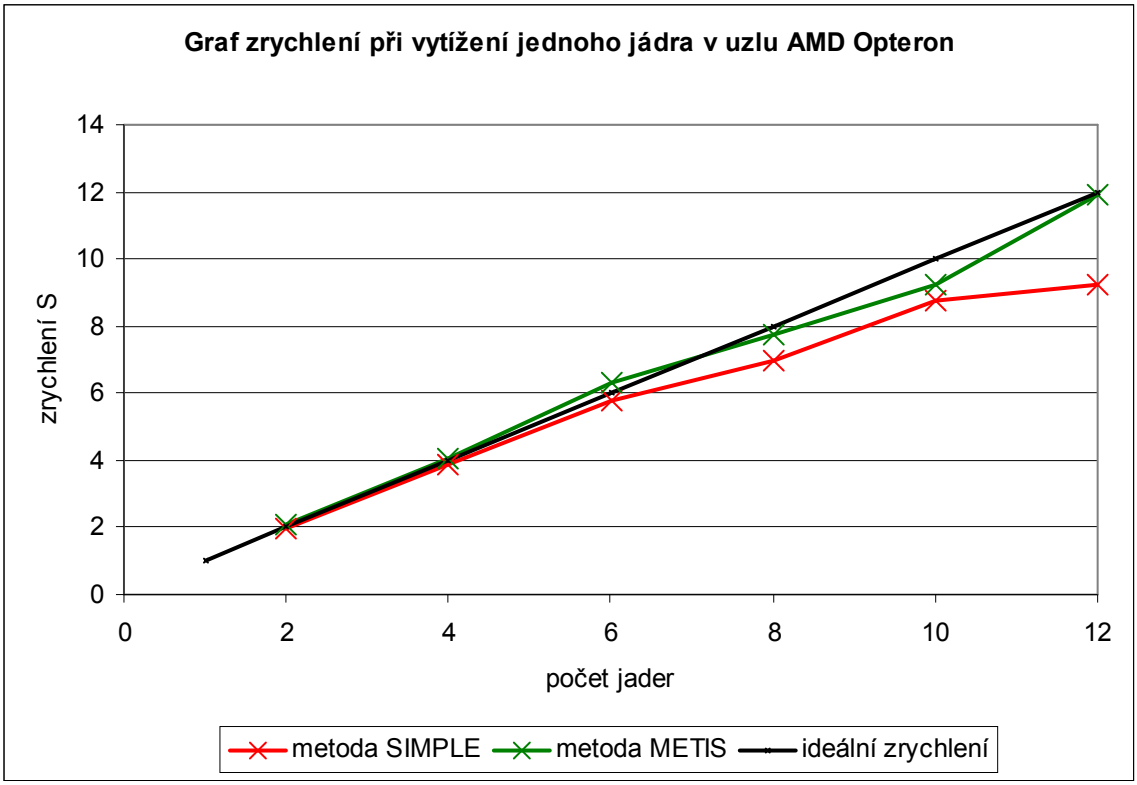
Graf 6.13: Zrychlení pro jednotlivé metody dekompozice při vytížení dvou jader v uzlu na uzlech Intel XEON



Graf 6.14: Zrychlení pro jednotlivé metody dekompozice při vytížení jednoho jádra v uzlu na uzlech Intel XEON



Graf 6.15: Zrychlení pro jednotlivé metody dekompozice při vytížení dvou jader z uzlu na uzlech AMD Opteron



Graf 6.16: Porovnání zrychlení pro jednotlivé metody dekompozice při vytížení jednoho jádra v uzlu na uzlech AMD Opteron

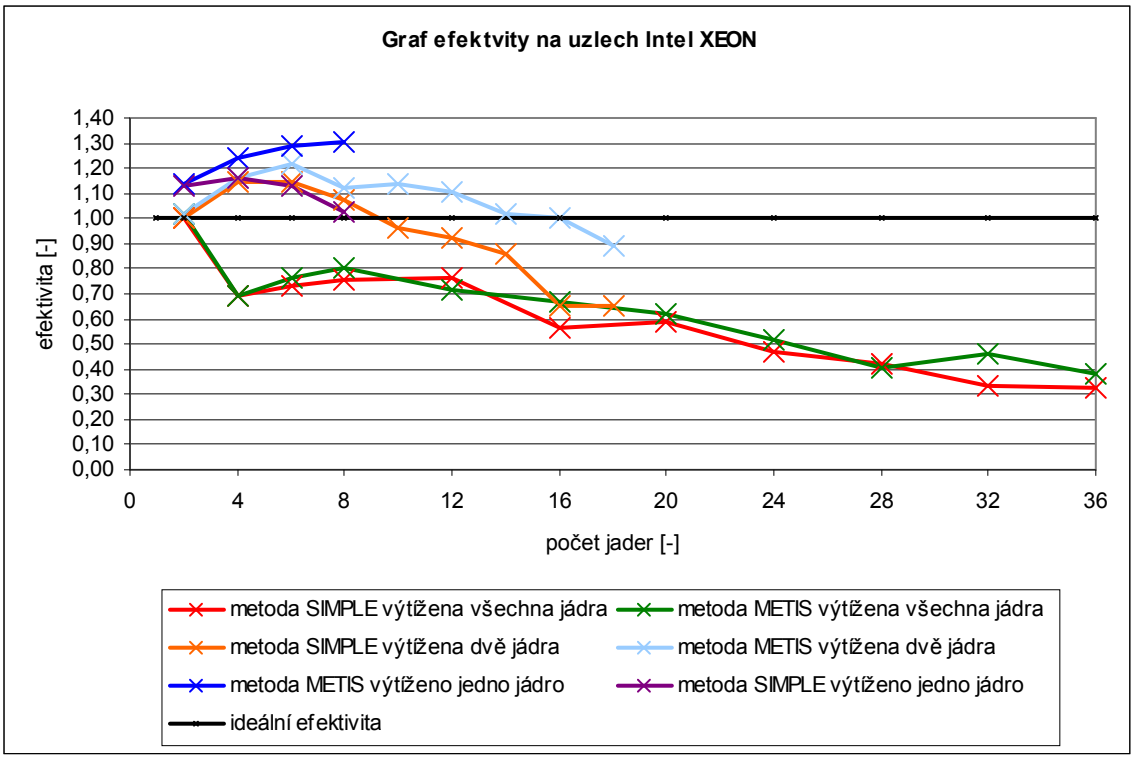
Poté, co jme ověřili, že *úloha 1* a *úloha 2* dosahují pro zrychlení podobné výsledky, je jasné, že stejný výsledek dostaneme i pro efektivitu. Největší efektivity je dosaženo pro oba typy uzlů při použití metody dekompozice METIS. U uzlů Intel XEON je efektivita výrazně ovlivněna vytížením uzlu. Opět se při plně vytíženém uzlu dosahuje velmi malé efektivity. Nejlepší pak při vytížení jednoho jádra. Jedinou výraznou odlišností je, že se u úlohy 2 dosahuje u uzlů AMD Opteron při použití metody SIMPLE menší efektivity než v případě úlohy 1.

Tab. 6.22: Efektivity pro jednotlivé metody dekompozice a vytížení uzlů, Intel XEON

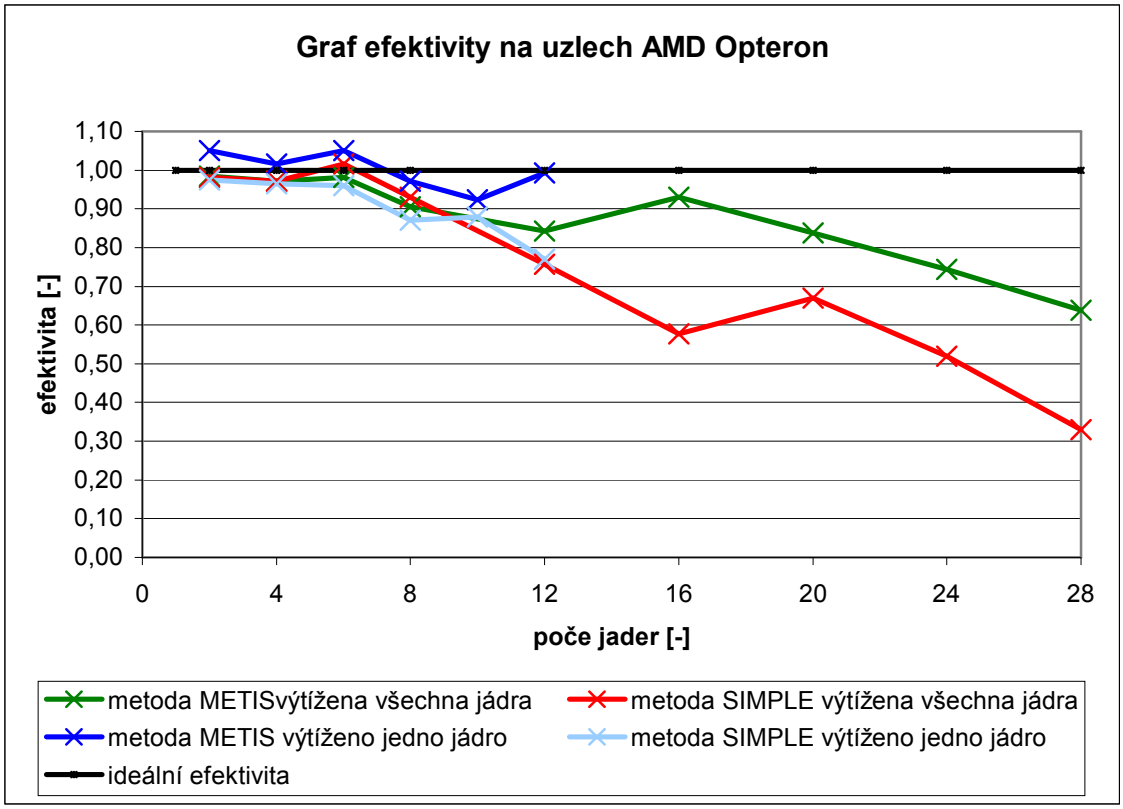
počet jader	efektivita Intel XEON					
	zatížena všechna jádra v uzlu		zatížena všechna jádra v uzlu		zatíženo jedno jádra v uzlu	
	METIS	SIMPLE	METIS	SIMPLE	METIS	SIMPLE
2	1,02	1,01	1,02	1,01	1,14	1,13
4	0,69	0,69	1,16	1,15	1,24	1,16
6	0,77	0,73	1,21	1,15	1,29	1,13
8	0,80	0,75	1,12	1,07	1,30	1,03
12	0,71	0,77	1,11	0,92	-	-
16	0,67	0,56	1,11	0,65	-	-
20	0,62	0,58	-	-	-	-
24	0,52	0,47	-	-	-	-
28	0,41	0,42	-	-	-	-
32	0,46	0,34	-	-	-	-
36	0,38	0,32	-	-	-	-

Tab. 6.23: Efektivity pro jednotlivé metody dekompozice a vytížení uzlů, AMD Opteron

počet jader	efektivita AMD Opteron			
	zatížena všechna jádra v uzlu		zatíženo jedno jádra v uzlu	
	METIS	SIMPLE	METIS	SIMPLE
2	0,99	0,98	1,05	0,97
4	0,97	0,97	1,02	0,96
6	0,98	1,02	1,05	0,96
8	0,91	0,93	0,97	0,87
12	0,84	0,76	0,99	0,77
16	0,93	0,58	-	-
20	0,84	0,67	-	-
24	0,74	0,52	-	-
28	0,64	0,33	-	-



Graf 6.17: Porovnání efektivity pro jednotlivé metody dekompozice a různém vytížení h uzlů na uzlech Intel XEON



Graf 6.18: Porovnání efektivity pro jednotlivé metody dekompozice a různém vytížení uzlů na uzlech AMD Opteron

6.5 Škálovatelnost paralelního výpočtu

Škálovatelnost je významným parametrem podle kterého se určuje, jak je úloha dobře paralelizovatelná. Významný vliv na škálovatelnost však může mít zvolený výpočetní paralelní algoritmus. Může se stát, že i při vhodné úloze k paralelním výpočtům můžeme špatným zvolením výpočetního algoritmu docílit špatné škálovatelnosti.

Po předchozích měření jsme zjistili, že výpočetní balík OpenFOAM disponuje kvalitním paralelním algoritmem pro řešič SIMPLE. Proto se nyní zaměříme na testovaní škálovatelnosti, při které se bude stejným poměrem zvětšovat objem vstupních dat a počet jader na kterých se bude výpočet spouštět. Zvětšování objemu dat se v tomto případě rozumí zvyšování počtu uzlů diskretizační sítě.

Pro rozklad oblasti byla zvolena metoda dekompozice METIS, výpočetní hardware jsem zvolil uzly Intel XEON při vytížení dvou jader. Tato kombinace byla zvolena záměrně, jelikož se při ní dosahovalo nejblíže ideálního zrychlení a efektivita se blížila jedné. Jako základní objem dat byla zvolena síť o 54225 uzlech a tato úloha se řešila na dvou jádrech. Ostatní úlohy pak měly *k* násobek uzlů základní úlohy, kde *k* bylo celé číslo. Při generování sítí nelze docílit vždy přesného *k* násobku počtu uzlů, bylo nutné výpočetní čas pomocí lineární interpolace přepočítat právě na hodnotu rovnou k násobku základní sítě. Za tímto účelem bylo pomocí programu GMSH vytvořeno devět sítí složených ze čtyřstěných elementů o různém počtu uzlů (viz tab. 6.24). Délka výpočtu byla stanovena na 20 iterací.

Po provedení výpočtu a zaznamenávání výsledku (viz tab. 6.24), se určila škálovatelnost, která je dána vztahem

$$\text{škálovatelnost}_{\text{k}} = \frac{T_{2*k}}{T_2}, \tag{6.3}$$

kde T_2 je výpočetní doba základní úlohy na dvou jádrech a T_{2*k} je výpočetní čas úlohy k krát větší než základní na $2k$ jádrech. V našem případě se jedná o dobře škálovatelnou úlohu a je tedy vhodná k paralelním výpočtům, jelikož i při velkém počtu jader se škálovatelnost blíží k hodnotě 1.5.

Tab. 6.24: Hodnoty získané při výpočtu škálovatelnosti

počet uzlů í sítě	počet elementů sítě	počet jader	využití paměti [MB]	doba výpočtu	čas přepočítaný lineární interpolací	škálovatelnost
54225	314501	2	126-159	31	31,00	1,00
105134	620367	4	128-165	33	34,04	1,10
160639	957296	6	121-149	37	37,47	1,21
206701	1238036	8	123-157	36	37,78	1,22
265498	1600350	10	120-160	42	42,89	1,38
328943	1988472	12	125-162	48	47,48	1,53
444295	2706299	16	125-176	49	47,84	1,54
483524	2943049	18	124-186	48	48,45	1,56

S ohledem na předešlé testování paralelních výpočtů se dá předpokládat, že se na hodnotě škálovatelnost projeví vliv jak metody dekompozice oblasti, tak i hardwaru. Z toho důvodu se tedy provedlo testování škálovatelnosti pro metodu dekompozice oblasti SIMPLE a výpočet se spouštěl na plně vytížených uzlech Intel XEON (viz tab. 6.25). Při této kombinaci metody dekompozice a hardwaru byly získány nejhorší výsledky efektivity a zrychlení. Zbytek vlastností jako výpočetní doba a typ elementů sítě byly ponechány stejné jako v prvním případě. Z tabulky je patrné, že hodnota škálovatelnosti vycházela výrazně větší než v prvním případě, více se zde projevila režie a latence.

Tab. 6.25: Hodnoty získané při výpočtu škálovatelnosti

počet uzlů výpočetní sítě	počet elementů výpočetní sítě	počet jader	využití paměti [MB]	doba výpočtu	čas přepočítaný lineární interpolací	škálovatelnost
54225	314501	2	126-159	31	31,0	1,00
105134	620367	4	128-165	54	55,7	1,80
160639	957296	6	121-149	59	59,7	1,93
206701	1238036	8	123-157	60	63,0	2,03
265498	1600350	10	120-160	62	63,3	2,04
328943	1988472	12	125-162	72	71,2	2,30

Závěr

Úkolem této diplomové práce bylo seznámení se s problematikou proudění tekutiny a numerickým řešením této problematiky. Poté ověřit možnost paralelního řešení a provést výpočet na úloze proudění tekutiny v měřící sekci experimentální trati pro sledování účinnosti filtrace. Právě paralelním výpočtům a jejich testování zrychlení a efektivity je věnována hlavní část této práce.

Prvním krokem tedy bylo nastudování literatury, která je věnována problematice proudění tekutiny a jeho numerickém řešené. Poté bylo provedeno numerické řešení úlohy proudění pomocí výpočetního balíku OpenFOAM. Před samotným numerickým výpočtem jsem musel namodelovat geometrii a vygenerovat výpočetní síť. K tomuto účelu byl použit open-source program GMSH. Všechny sítě generované pro potřeby numerického výpočtu byly složené z čtyřstěnných prvků.

Pro numerické řešení proudění v nádobě byl zvolen výpočetní balík OpenFOAM. Jedná se open-source balík objektově orientovaných výpočetních knihoven naprogramovaný v jazyce C++ s možností zásahu do výpočetního kódu. Tento balík umožňuje numerické řešení celé řady fyzikálních úloh, zejména z oblasti proudění tekutin. Pro tuto problematiku je vybaven rozsáhlým souborem řešičů jak pro stlačitelné tak nestlačitelné proudění tekutiny. V našem případu se vybíral rešič pro nestlačitelné proudění a to i přesto, že se jedná o proudění plynu. Toto si můžeme dovolit pouze v případě malých rychlostí, kdy je splněna podmínka $u<0,3Ma$.

Z celého spektra řešičů pro nestlačitelné proudění byl zvolen simpleFOAM. Jedná se o stacionární solver, který počítá ustálený stav proudění nestlačitelné tekutiny.

Po úspěšném provedení sekvenčního výpočtu a ověření správnosti výsledku následovalo převedení úlohy ze sekvenčního výpočtu na výpočet paralelní. Paralelní výpočet je oproti sekvenčnímu výrazně složitější a vyžaduje celou řadu úkonů (rozklad oblasti, paralelní výpočetní zařízení a výběr paralelního solveru). Pro paralelní výpočty byl stejně jako pro sekvenční výpočty použit výpočetní balík OpenFOAM, který pro tuto problematiku nabízí vhodné nástroje. Pro rozklad oblasti nabízí několik metod dekompozice, použity byly dvě a to metody SIMPLE a METIS. Hlavní rozdíl je ve způsobu rozkladu, metoda SIMPLE dělí oblast na podoblastí se stejný počet elementů, zatímco metoda METIS provádí rozklad tak, aby byly nároky na komunikaci během výpočtu co nejmenší. Paralelní výpočty se počítaly na školním výpočetním clusteru Hydra. Jedná se o heterogení cluster, který je složený ze dvou typů výpočetních uzlů. Výpočetní uzly jsou založeny na platformě AMD Opteron a Intel XEON (viz tab. 6.1).

Poté následovalo testování paralelních výpočtů, testovalo se zrychlení a efektivita paralelního výpočtu. Testování se provádělo pro jednotlivé metody rozkladu oblasti i pro oba typy výpočetních uzlů, kde se navíc testovalo i obsazení jader v uzlu. U uzlů AMD Opteron se spouštěl výpočet na jednom a na dvou jádrech v uzlu, jelikož je uzel tvořen dvěmi jednojádrovými procesory. U uzlů Intel XEON se pak spouštěl na čtyřech, dvou a jednom jádru v uzlu, jelikož je uzel tvořen dvěmi dvoujádrovými procesory. Pro ověření výsledků se testování provádělo na dvou samostatných úlohách. Úlohy se lišily ve velikosti sítě, úloha jedna 1 byla tvořena 3,25 miliony elementů a úloha 2 pak 957 tisíc elementů. Pro obě úlohy byly získány přibližně stejné výsledky.

Ze všech výpočtů se zjistilo, že lepšího zrychlení a efektivity se dosahovalo při použití metody dekompozice METIS. U metody SIMPLE se pro řešení na více jádrech začala výrazněji projevovat režie výpočtů vlivem horší dekompozice.

K určitému překvapení pak došlo, když se porovnávalo zrychlení a efektivita v závislosti na obsazení výpočetního uzlu. U uzlů Intel XEON se nejhoršího výsledku dosahovalo při obsazení všech jader v uzlu. Naopak pokud se vytěžovalo pouze jedno jádro bylo zrychlení výpočtu větší než lineární a dosahovalo se při výpočtu na 8 jádrech zrychlení 10,4. To je nejspíše způsobeno tím, že uzly Intel XEON mají sdílenou L2-cache a pokud máme k výpočtu použita obě jádra v procesoru, musí obě jádra využít více paměť RAM, která je oproti paměti cache pomalejší. Další vliv na zrychlení má také samotná paměť RAM, protože uzel Intel XEON využívá pro oba procesory jednu sdílenou paměť.

Velký rozdíl mezi oběma uzly je v přístupu do paměti. Uzly AMD Opteron, mají přístup do paměti NUMA (Non-Uniform Memory Access), každý procesor má vlastní přístup do paměti RAM, tím pádem se navzájem neomezují. Vliv vytížení jader se tak výrazně neprojevil, jako v případě uzlu Intel XEON (byl velmi malý).

Zhodnocením výsledku však práce na tomto tématu není uzavřena, proto se ve svém dalším studiu budu snažit tuto problematiku dále rozvést. Bylo by také vhodné dosažené výsledky ověřit i na jiném výpočetním clusteru, aby bylo možné najít přesné příčiny rozdílného zrychlení výpočtu, které byli získaný při výpočtech na clusteru Hydra. Jelikož tato práce má být postupem času využita v praxi musí se stávající úloha rozšířit o podporu výpočtu proudění přes samotný filtr a najít vhodnější nástroj pro generování sítí. Nakonec rozšířit výpočet ze stávajícího laminárního modelu proudění na turbulentní model.

Seznam použité literatury

[1] BEJCHAŘ, Tomáš. Turbulence Modelování proudění-CFX [online]. Vysoká škola bánská-Technická univerzita Ostrava, 2008. [cit. 12. prosince 2010] URL: <http://www.338.vsb.cz/PDF/Turbulence_ESF_v4.pdf>

[2] FOŘT Jaroslav; KOZEL Karel; LOUDA Petr; FÜRST Jiří. NUMERICKÉ METODY ŘEŠENÍ PROBLÉMŮ PROUDĚNÍ III. 1. vyd. Praha: České vysoké učení technické v Praze, 2004. 95 s. ISBN 987-80-01-02877-1

[3] JAKL, Ondřej. Paralelní systémy 2005[online]. [cit. 13. března 2011] URL: <http://www.cs.vsb.cz/jakl/pds/pds_lec.pdf>

[4] JANALÍK, Jaroslav. HYDRODYAMIKA A HYDRODYNAMICKÉ STROJE [online]. Vysoká škola bánská-Technická univerzita Ostrava, 2008. [cit. 1. února 2011] URL: <http://www.338.vsb.cz/PDF/Janalik-HYDRODYNAMIKAA HYDRODYNAMICKESTROJE.pdf>

[5] KOZUBKOVÁ, Milada. Modelování proudění tekutin FLUENT, CFX [online]. Vysoká škola bánská-Technická univerzita Ostrava, 2010. [cit. 12. prosince 2010] URL: < http://www.338.vsb.cz/PDF/Kozubkova-Fluent.pdf>

[6] NOŽIČKA, Jiří. MECHANIKA TEKUTIN. 1. vyd. Praha: České vysoké učení technické v Praze, 2004 165 s. ISBN 80-01-02865-8

[7] PŘÍHODA, Jaromír; LOUDA Petr. MATEMATICKÉ MODELOVÁNÍ TURBULENTNÍHO PROUDĚNÍ. 1. vyd. Praha: České vysoké učení technické v Praze, 2009. 111 s. ISBN 987-80-01-03623-5

[8] ŠEMBERA, J. MECHANIKA TEKUTIN [online]. Technická univerzita v Liberci, 2004 [cit. 15. ledna 2011] URL:<http://www.nti.tul.cz/cz/images/e/e9/METskr.pdf>

[9] URUBA, Václav. TURBULENCE. 1. vyd. Praha: České vysoké učení technické v Praze, 2009. 130 s. ISBN 987-80-01-04330-1

[10] WHITE, F. M. Fluid Dynamics, McGraw-Hill, 2006.

[11] Gmsh Reference Manual The documentation for Gmsh 2.4 A finite element mesh generator with built-in pre- and post-processing facilities 18. 10. 2009

[12] Open FOAM *The Open Source CFD Toolbox* User Guide [online] Version 1.7.1 25th August 2010 [cit. 6. dubna 2011] URL: <http://foam.sourceforge.net/docs/Guides-a4/UserGuide.pdf>

[13] Domain decomposition methods [online] [cit. 10.března 2011] URL: <http://en.wikipedia.org/wiki/Domain_decomposition_methods>

[14] Detailní technické informace o clusteru Hydra [online] [cit. 10. února 2011]
URL: <http://www.nti.tul.cz/cz/Hydra >

[15] Parallel computing [online] [cit. 6. ledna 2011]
URL: <http://en.wikipedia.org/wiki/Parallel_computing >

[16] NVIDIA CUDA [online] [cit. 10.března 2011]
URL: <http://www.nvidia.com/object/cuda_home_new.html>

[17] Computational fluid dynamics [online] [cit. 5. března 2011]
URL: <http://en.wikipedia.org/wiki/Computational_fluid_dynamics>

[18] A Parallel Implementation of the BDDC Method for the Stokes Flow [online] [cit. 19 dubna 2011]
URL: < http://www.math.cas.cz/~sistek/talks/Sistek-2010-ICCFD-talk.pdf>

Příloha
Příloha A

soubor s okrajovými podmínkami pro tlak

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    Gin
    {
        type            zeroGradient;
    }

    Gout
    {
       type           fixedValue;
       value          uniform 0.0;
    }

    Wall
    {
        type            zeroGradient;
    }

}

      // ************************************************************************* //
```

soubor s okrajovými podmínkami pro rychlost

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      http://www.OpenFOAM.org               |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    Gin
    {
        type            fixedValue;
        value           uniform (0.1 0 0);
    }

    Gout
    {
         type         zeroGradient;
    }

    Wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

}

// ************************************************************************* //
```

Příloha B

-soubor s metodou dekompozice METIS

-je zde uveden příklad pro rozklad na 10 podoblastí, kde všechny podoblasti mají stejnou váhu

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 10;

method          metis;

metisCoeffs
{
    processorWeights ( 1 1 1 1 1 1 1 1 1 1 );
    delta             0.0001;
    strategy;
}

distributed     no;

roots           ( );

// ************************************************************************* //
```

Příloha C

-soubor s metodou dekompozice SIMPLE

-je zde uveden příklad pro rozklad na 10 podoblastí, metoda dělí oblast ve směru souřadné osy z

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.7.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 10;

method              simple;

simpleCoeffs
{
    n               ( 1 1 10 );
    delta           0.00001;
}



distributed     no;

roots           ( );

// ************************************************************************* //
```

Příloha D

soubor se seznamem uzlů AMD Opteron

```
compute-1-16.local cpu=2
compute-1-15.local cpu=2
compute-1-14.local cpu=2
compute-1-13.local cpu=2
compute-1-12.local cpu=2
compute-1-11.local cpu=2
compute-1-10.local cpu=2
compute-1-9.local cpu=2
compute-1-8.local cpu=2
compute-1-7.local cpu=2
compute-1-6.local cpu=2
compute-1-5.local cpu=2
compute-1-4.local cpu=2
compute-1-3.local cpu=2
compute-1-2.local cpu=2
compute-1-1.local cpu=2
compute-1-0.local cpu=2
```

soubor se seznamem uzlů Intel XEON

```
compute-0-0.local cpu=4
compute-0-1.local cpu=4
compute-0-2.local cpu=4
compute-0-3.local cpu=4
compute-0-4.local cpu=4
compute-0-5.local cpu=4
compute-0-6.local cpu=4
compute-0-7.local cpu=4
compute-0-9.local cpu=4
compute-0-10.local cpu=4
```