IMPLEMENTIMI PARALEL PËR EKUACIONIN PARABOLIK NË DY PLATFORMA

MENTOR SHEVROJA.

Universiteti i Tiranës, , Fakulteti i Shkencave të Natyrës, Departamenti i Matematikës së Aplikuar, (Morix shpk, Qendra Condor, III/15)

e-mail: mentor.shevroja@fshn.edu.al

Përmbledhje

Në këtë punim do të paraqesim implementimin paralel për zgjidhjen numerike të ekuacionit parabolik duke përdorur metodën e diferencave të fundme, në platformat OpenMP dhe programimin paralel në gjuhën C# për të njëjtën madhësi të problemit në të dy platformat. Testet numerike janë kryer për të gjetur platformën ku paralelizimi është më efektiv në dy aspekte *kohën e njehsimit paralel* dhe *kohën e krijimit të fijeve*. Implementimet janë orientuar në drejtim të programimit me performancë të lartë dhe për probleme me përmasa të mëdha. Disa nga temat e tjera të diskutuara janë kostoja e krijimit të fijeve, sinkronizimi i tyre, aksesi tek të dhënat e përbashkëta në programimin paralel ku janë zhvilluar algorimet në dy platformat e sipërpërmendura.

Fjalëkyçe: Programim paralel, OpenMP, gjuha C#, ekuacioni parabolik, multithreading.

Abstract

In this paper we represent a parallel implementation for numerical solution of the parabolic equation using finite difference method, using OpenMP (Open Multi-Processing) and Multithreaded C# for the same problem size. The numerical tests are performed to find the environment where parallelization is more effective in time of computation and thread time creation. The implementations are oriented on high performance computation and big-data. Also some other topics discussed are thread creation, thread synchronization and data race in parallel programming.

Key words: Parallel programming, OpenMP, C#, parabolic equation, multithreading.

Hyrje

Në këtë material do të paraqesim implementimin paralel për zgjidhjen numerike të ekuacionit parabolik duke përdorur metodën e diferencave të fundme, në platformat OpenMP dhe programimin paralel në gjuhën C#. Testet numerike janë kryer për të gjetur platformën ku paralelizimi është më efektiv në dy aspekte *kohën e njehsimit paralel* dhe *sinkronizimin e fijeve*. Implementimet janë orientuar në drejtim të programimit me performancë të lartë dhe për probleme me përmasa të mëdha. Disa nga temat e tjera të diskutuara janë kostoja e krijimit të fijeve, sinkronizimi i tyre, aksesi tek të dhënat e përbashkëta në programimin paralel ku janë zhvilluar algorimet në dy platformat e sipërpërmendura.

Ekuacionet parabolike diferenciale te pjesshme zakonisht përdoren në fusha të tilla si difuzioni molekular, transferimi i nxehtësisë, analiza e reaktorit bërthamor dhe rrjedha e lëngjeve. Ekuacionet diferenciale me derivate të pjesshme përdoren gjerësisht si modele matematikore për fenomenet në të gjitha degët e inxhinierisë dhe shkencës.

Programimi paralel është një formë e kryerjes llogaritjeve në disa proçesorë njëkohësisht, i cili përdoret gjerësisht për të kursyer kohë dhe para, për të zgjidhur probleme që kërkojnë shumë veprime, për të përdorur burime në largësi etj. Programimi paralel me fije (multithreading) është një model i gjerë i programimit dhe ekzekutimit që lejon fijet e shumta të ekzistojnë brenda kontekstit të një procesi. Këto fije i ndajnë burimet e procesit dhe janë në gjëndje të ekzekutojnë metoda në mënyrë të pavarur.

Zgjidhja numerike nëpërmjet diferencave të fundme

Metoda e diferencave të fundme (Hoxha, 2008) është një mënyrë klasike dhe e drejtpërdrejtë për ti zgjidhur ekuacionet diferenciale me derivate të pjesshme numerikisht . Ekuacioni parabolik ka këtë formë

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \quad \text{në} \quad D: \begin{cases} 0 \le x \le L \\ 0 \le t \le T \end{cases} \quad (1)$$

Me kushtin fillestar $u(x,0) = F(x),\ 0 \le x \le L$ dhe kushte kufitare

$u(0,t) = \varphi(y),\ u(L,t) = \psi(y), 0 \le t \le T$. Ku $F(x), \varphi(y), \psi(y)$ janë funksione të njohura.

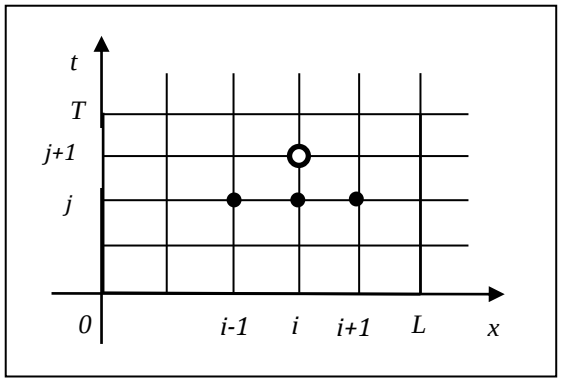

Figura 1. Zgjidhja numerike me motoden e shtjellur

Zgjidhja numerike nëpërmjet diferencave të fundme konsiston në ndajen e zonës D me anë të dy familjeve drejdëzash paralele me boshtet koordinative, $x_i = a + i * h, y_i = a + j * k$ ku h është hapi sipas boshtit x dhe k është hapi sipas

boshtit y. Si rrjedhim, formohet një rrjet pikash, që janë pikëprerje e këtyre dy familjeve drejdëzash të paraqitura në figurën 1.

Shënojmë $u(x_i, y_j) = u_{ij}$ për $i = \overline{1 \dots n}, j = \overline{1 \dots m}$. Në çdo pikë (x_i, y_j) të brendshme të zonës D, zëvëndësojmë derivatet e pjesshme me diferencat e fundme. Për u'_t përdorim diferencat e fundme të përparme, ndërsa për $\frac{\partial^2 u}{\partial x^2}$, diferencat e fundme qëndrore.

$$\left(\frac{\partial u}{\partial t}\right)_{ij} \approx \frac{u_{ij+1} - u_{ij}}{k}, \qquad \left(\frac{\partial^2 u}{\partial x^2}\right)_{ij} \approx \frac{u_{ij+1} - 2u_{ij} + u_{i+1j}}{h^2}$$

Duke zëvëndësuar këto vlera tek ekuacioni (1) kemi

$$\frac{u_{ij+1} - u_{ij}}{k} = \frac{u_{ij+1} - 2u_{ij} + u_{i+1j}}{h^2} \iff u_{ij+1} = u_{ij} + \frac{k}{h^2}\left(u_{i-1\,j} - 2u_{ij} + u_{i+1\,j}\right)$$

Në qoftë se shënojmë $\alpha = \frac{k}{h^2}$, atëherë ekuacioni i mesipërm rishkruhet

$$u_{ij+1} = \alpha\, u_{i-1\,j} + (1 - 2\alpha)\, u_{ij} + \alpha u_{i+1\,j}) \quad (2)$$

Duke ditur funksionet e kushteve fillestare dhe ato kufitare, ne mund të njehsojmë vlerat e tjera të panjohura të funksionit të panjohur $u(x, t)$ në zonën *D*. Vërtetohet që skema (2) është e qëndrueshme (Hoxha 2008) kur $0 \leq \alpha \leq 0.5$.

Pseudokodi në paralel

Pseudokodi për paralelizim mund të përmblidhet në këto hapa:

1. Lexo numrin e fijeve *s*, *h*, *k*.
2. Fillo proçeso në paralel me *s* fije
- 2.1. Llogarit kushtet fillestare
- 2.2 Për *j* nga *1* në *m* bëj
- 2.2.1 Sinkronizo fijet
- 2.2.2 Kopjo vektorin *u* tek *v*
- 2.2.3 Sinkronizo fijet
- 2.2.4 Llogarit vlerat e reja të u në iteracionin *j+1*
- 2.3 Fund për *j*
3. Shfaq rezultatet dhe përfundo

Siç shihet edhe nga pseudokodi, dy hapat e rëndësishëm janë sinkronizimi dy herë gjatë çdo iteracioni j. Pra fijet duhet të pozicionohen në atë pikë në mënyrë që të sigurohemi se të gjitha llogaritjet janë kryer para se të kalojmë në hapin tjetër.

Varianti i algoritmit që është përdorur ka si qëllim përdorimin e dy vektoreve *u* dhe *v*, ku vektori *u* përfaqëson vlerat ne iteracionin e fundit dhe vektori *v* përfaqëson vlerat në iteracionin e parafundit. Algoritmi me matricë nuk është përdorur pasi nuk është e mundur të deklarohen matrica me përmasa *1 000 000 x 480* në mënyrë që të njehsohen vlerat e funksionit në nyjet e brendshme të rrjetës u_{ij}. Duke përdorur vektorët me përmasa të mëdha mund të njehsohen me lehtësi dhe pa shumë memorje në dispozicion dhe mund te shkruajmë në skedarë rezultatet e çdo iteracioni.

Ky lloj ndryshimi ka një të keqe sepse na duhet të sinkronizojmë fijet në dy etapa për çdo iteracion. Momenti i parë i sinkronizimit është kopjimi i vlerave të *u* tek *v* dhe momenti i dytë është para se të fillojmë njehsimin e vlerave të vektorit *u* duke u siguruar që vlerat e vektorit *v* janë përditësuar. Ky sinkronizim duhet të bëhet në çdo nivel *j* dhe po të mendosh që për një hap *k* të vogël, numri i sinkronizimeve bëhet goxha i lart, kjo sigurisht që ndikon në performacë, por nuk mund të evitohet për një ndarje të imët të boshtit të kohës *ot*.

Implementimet paralele

Janë zhvilluar algoritmat paralel në dy platforma. Implementimi i parë është kryer në OpenMP/C++, ndërsa implementimi i dytë është kryer në .NET/C# ambjent programimi i avancuar me objekte për programim sekuencial dhe paralel.

Në mënyrë që të verifikohen rezultatet që prodhojnë algoritmet paralele, rezultatet janë krahasur me zgjidhjen numerike të një ekuacioni parabolik të njohur:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + 2 - 6x \ \text{në} \ D: \begin{cases} 0 \le x \le 1 \\ 0 \le t \le 2 \end{cases}$$

Me kushte fillestare dhe kufitare $\begin{cases} u(x,0) = \sin(\pi x) + x(1-x)^2, \ 0 \le x \le 1 \\ u(0,t) = 0, \ u(1,t) = 0, t \ge 0 \end{cases}$

Ndarja e boshteve është bërë në mënyrë që të përftohen rezultate të qëndrueshme (Hoxha, 2008) nga metoda e diferencave të fundme $h = \frac{1}{480}, k = \frac{1}{1000000}, \alpha = \frac{k}{h^2} = 0.2304 < 0.$

Ku zgjidhja e saktë e këtij ekuacioni parabolik është $u(x,t) = e^{-\pi^2 t}\sin(\pi x) + x(1-x)^2$.

Janë kryer implementimet për zgjidhjen e ekuacionit parabolik dhe janë testuar në të njejtën makinë. Të gjitha testet janë kryer në Windows Server 2012 R2, me proçesor Intel Xeon E5-2420, 12 Core fizike, 24 Core virtuale, secili me shpejtësi 1.9GHz.

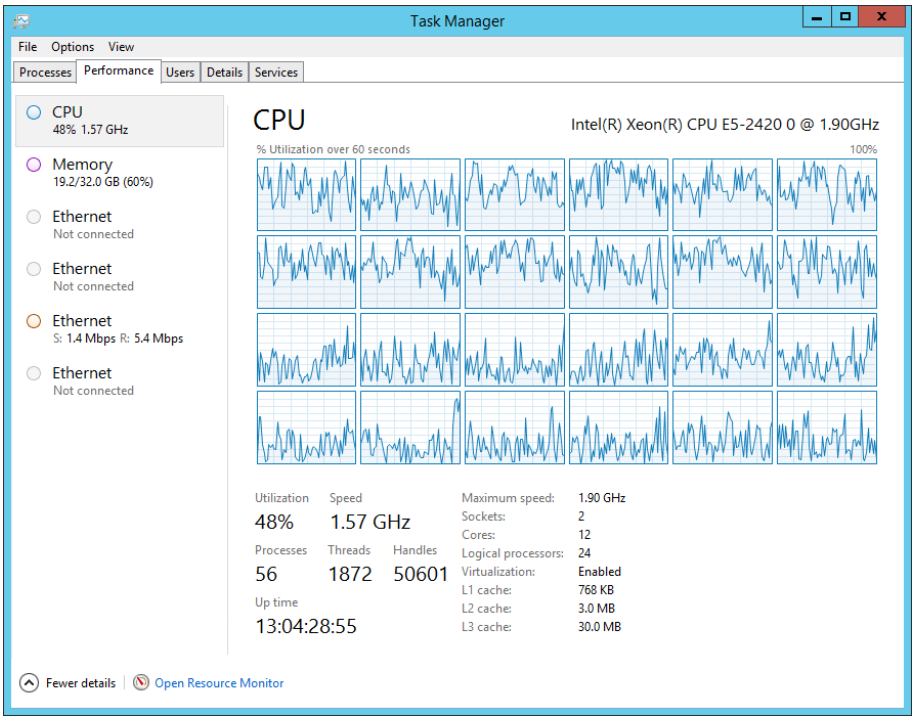

Figura 1. Foto të parametrave të serverit gjatë ekzekutimit të algorimeve

Ekzekutimet janë kryer gjatë një ditë të zakonshme pune, ndërkohë që makina ka qenë në shërbim të detyrave të tjera të paracaktuara. Duke qënë se serveri duhet të shërbej 24 orë për atë çfarë pëdroet, algoritmet tona nuk mund të ekzekutoheshin vetëm por me programet në perdorim. Megjithatë për kryerjen e implementimit është zgjedhur dita kur ka pasur më pak ngarkesë në mënyrë që të shfrytëzohen sa më shumë kapacitetet e makinës.

Për rezultatet janë mbledhur dhe llogaritur disa vlerësues të paralelizimit:

- T_p - koha e ekzekutimit të algorimit në *p* proçesorë (Hoxha, 2007).
- Shpejtësia (speedup) $S_p = \frac{T_1}{T_p}$ e ku T_1 – koha e njehsimit më një proçesor dhe T_p – koha e njehsimit me *p* proçesorë (Hoxha, 2007).

- Efektshmëria $E_p = \frac{S_p}{p}$ është raporti shpejtësisë me p proçesorë me numrin e proçesorëve (Hoxha, 2007).

- Efektiviteti $F_p = \frac{E_p}{T_p}$ është raporti i i Efektshmerisë me kohën e njehsimit me p proçesorë. Ndryshe mund të shkruhet $F_p = \frac{S_p}{pT_p} = \frac{E_p S_p}{T_1}$ (Hoxha, 2007).

Në tabelat e mëposhtme paraqiten rezultatet e testimeve të kryera në dy platformat për rastet me algoritmin sekuecial dhe paralel

Nr fijeve	T_p	S_p	E_p	F_p
1	24.6460	-	-	-
2	17.8070	1.38406	0.69203	0.05616
4	15.4210	1.59821	0.39955	0.06485
8	13.6910	1.80016	0.22502	0.07304
12	13.5680	1.81648	0.15137	0.07370
16	14.2820	1.72567	0.10785	0.07002
24	14.8430	1.66045	0.06919	0.06737
32	15.8670	1.55329	0.04854	0.06302

Tabela 1. Treguesit e paralelizimit për rastin e platformës C++/OpenMP

Nr fijeve	t_p	S_p	E_p	F_p
1	26.5601	-	-	-
2	19.9881	1.32880	0.66440	0.05003
4	16.4139	1.61815	0.40454	0.06092
8	14.9275	1.77927	0.22241	0.06699
12	13.7024	1.93835	0.16153	0.07298
16	13.7634	1.92976	0.12061	0.07266
24	14.1783	1.87329	0.07805	0.07053
32	14.4537	1.83760	0.05742	0.06919

Tabela 2. Treguesit e paralelizimit për rastin e platformës C#/.NET

Rezultatet e tableva të mësipërme janë analizuar dhe më poshtë do gjeni grafikët e vlerësuesve të paralelizimit për të dy platformat.

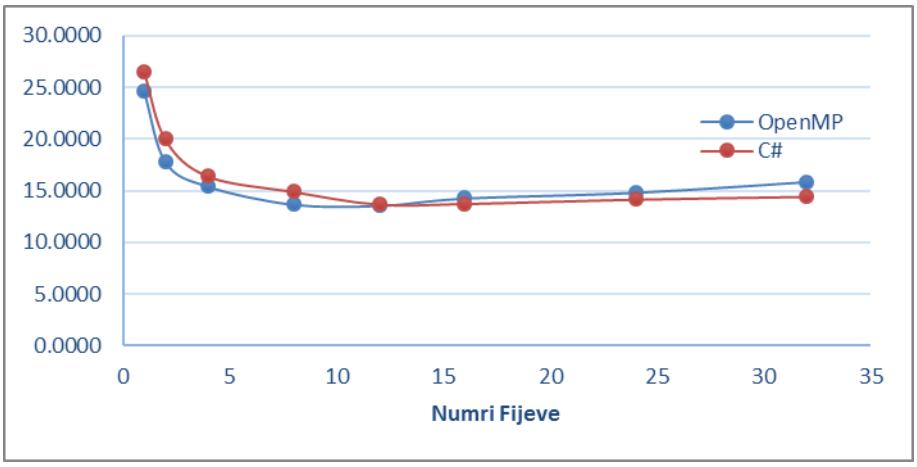

Figura 2. Koha e ekzekutimit të ekuacionit parabolik

Mund të shohim se C++ kryen më shumë njehsime në njësinë e kohës si në sekuecor ashtu edhe në rastin kur numri i fijeve nuk e kalon 12. Nga ana tjetër C# tregohet më i shpejtë në rastin e njehsimit paralel me më shumë se 12 fije.

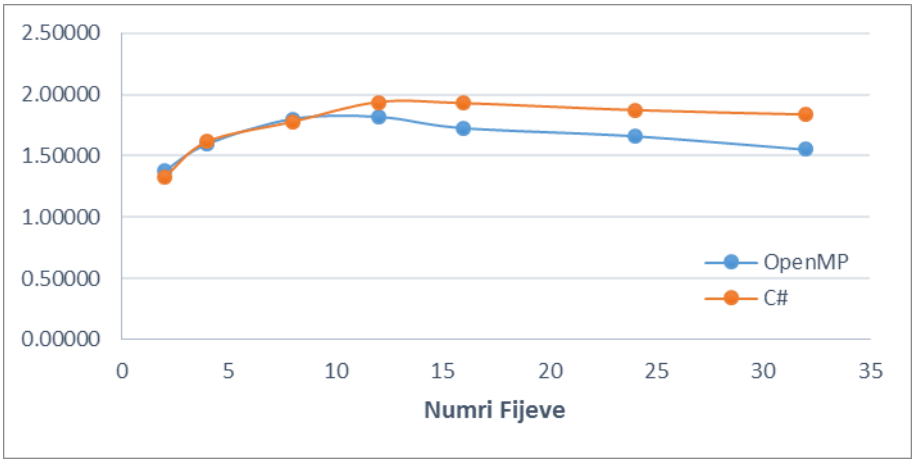

Figura 3. Shpejtësia (speedup) e algoritmave në varësi të numrit të fijeve

Nga Figura 4 duket qartë se të dy platformat kanë shpejtesi të njejtë përsa i takon algoritmeve sekuencor dhe atyre paralel deri në 12 fije. Me rritjen e numrit të fijeve, C# tregohet pak më i shpejt. Kjo do të thotë se C# i kryen me efikasitet sinkronizimet dhe ndarjen e detyrave të fijeve në proçesorë.

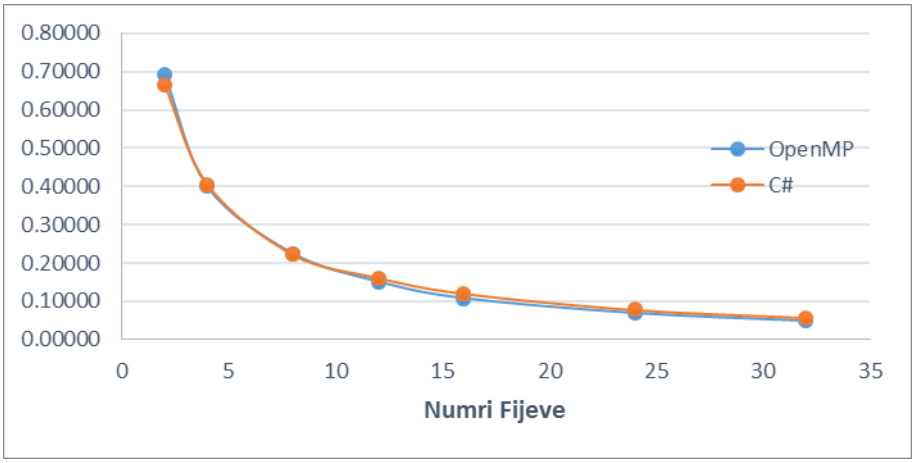

Figura 4. Efektshmëria (efficency) e problemit në varësi të numrit të fijeve

Efektshmëria është thuajse e njejtë. OpenMP ka një përparësi të vogël për rastin e problemit me më pak se 12 fije.

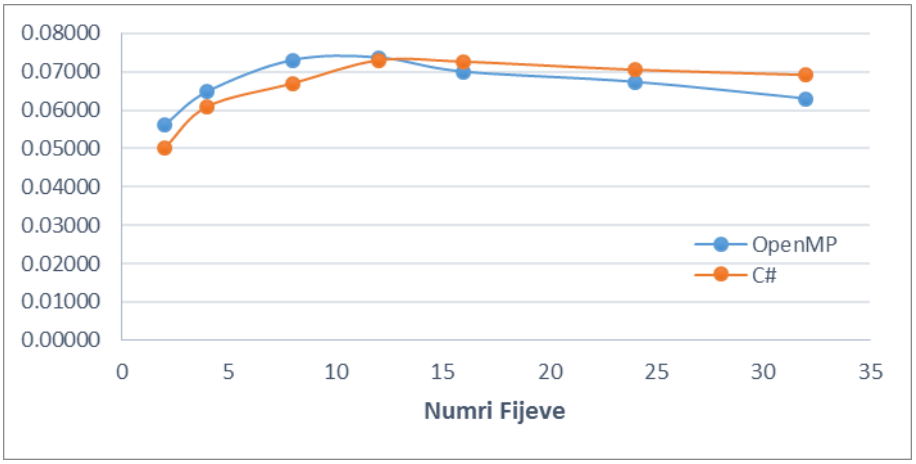

Figura 5. Efektiviteti (Effectivness) në varësi të numrit të fijeve

Shihet qartë nga Figura 6 se OpenMP ka efektivitet më të mirë sesa C#. Kjo do të thotë se gjatë punës në paralel C++ kryen më shumë njehsime në paralel sesa C#. Kjo është e vërtetë për rastet kur nuk tejkalohet numri fizik i proçesorëve.

Përfundime

Metodat numerike në zgjidhjen e problemeve në paralel ka një rëndësi në rritje çdo ditë edhe më shumë, edhe kompjuterat personal prodhohen me shumë proçesorë në mënyrë që të kryejnë njehsime paralele me fije.

Nga rezultatet e paraqitura shihet qartë se OpenMP ka përparësi ndaj .NET në rastet nuk numri i fijeve nuk e kalon numrin e proçesorëve fizik të makinës. Nga ana tjetër .NET ka performancë më të mirë në rastet kur numri i fijeve tejkalon numrin e proçesoreve fizik të makinës.

Përdorimi i matricave në paralelizimin e ekuacionit parabolik nuk do ishte efektiv pasi do kërkonin memorie të madhe për ndarje të vogla të h dhe k, pandaj përdorimi i vektorëve ul ndjeshëm kërkesën për memorie.

Për probleme me performancë të lartë, nuk ka arsye pse problemet të ndahen në më shumë proçesorë sesa mbart makina pasi kjo ndikon në uljen e kohës totale së zgjidhjes së problemit.

Literatura

Hoxha F. (2007): Njehsim Paralel: 67-75

Hoxha F. (2008): Metoda të Analizës Numerike: 199-208

Benito J.J, Urena F., Gavete L. (2007): Journal of Computational and Applied Mathematics, Solving parabolic and hyperbolic equations by the generalized finite difference method: Volume , Issue 2: 208-233

Larsson S., Thomee V. (2003): Partial Differential Equations with Numerical Methods, Finite Difference Methods for Parabolic Problems 129-148