**Lygiagretusis programavimas su OpenMP**

Panaudodami OpenMP technologiją, parašykite programą, vykdančią lygiagrečius skaičiavimus.

1. Sukurkite tris matricas. Pirmoji – n eilučių ir k stulpelių. Antroji – k eilučių ir p stulpelių. Trečioji – n eilučių ir p stulpelių.
2. Atlikite veiksmus su matricomis:
   1. Pirmosios matricos elementams priskirkite atsitiktinai sugeneruotus sveikus skaičius (galimos reikšmės - nuo 1 iki x). Šią programos dalį vykdykite lygiagrečiai (su pasirinktu gijų skaičiumi).
   2. Antrosios matricos elementams atsitiktine tvarka priskirkite 1 arba -1. Šią programos dalį vykdykite nuosekliai.
   3. Sudauginkite pirmą ir antrą matricas, gautus rezultatus priskirkite trečiai matricai. Šią programos dalį vykdykite lygiagrečiai (su pasirinktu gijų skaičiumi).
3. Išveskite rezultatus. Išanalizuokite, kaip keičiasi programos vykdymo laikas, priklausomai nuo:
   1. Naudojamų branduolių skaičiaus
   2. Matricų dydžių

Pateikite analizės rezultatus (kartu su „printscreenais“, iliustruojančiais procesoriaus branduolių apkrovimą programos vykdymo metu).

Antanas Valenčius - ISI 1k. 2gr. 2pgr.

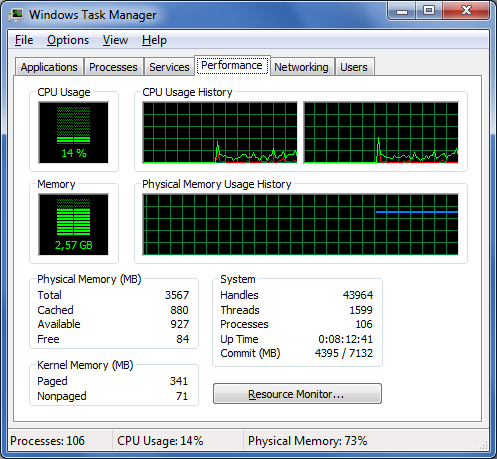
**Įžanga:**  
Pasinaudodamas šablonu bei <https://computing.llnl.gov/tutorials/openMP/> puslapiu bandžiau išsiaiškinti lygiagretaus programavimo ypatybes bei metodus. Puslapyje yra paaiškinimai, kurie šiek tiek glumina, bet ir pavyzdžiai ir užduotys, kurios padeda susiorientuoti, kur kokia komanda ką daro.

**Tikslas:**  
Mano tikslas buvo pasinaudoti šablonu ir bent suprasti, kokia nauda ir trūkumai yra iš lygiagretaus programavimo ir kaip tą naudą įgyti.

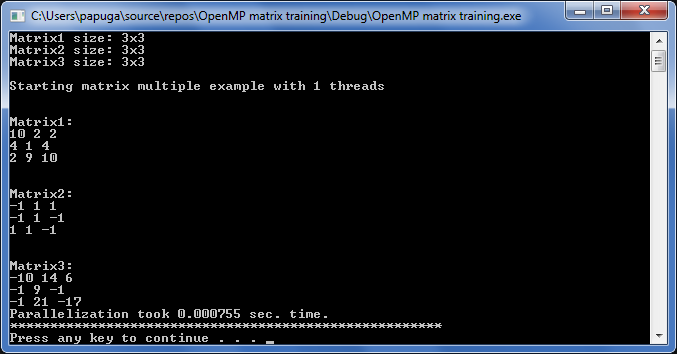
Pridėtame „OpenMP matrix training.cpp“ faile yra programos kodas, kuriuo toliau vadovausiuos.

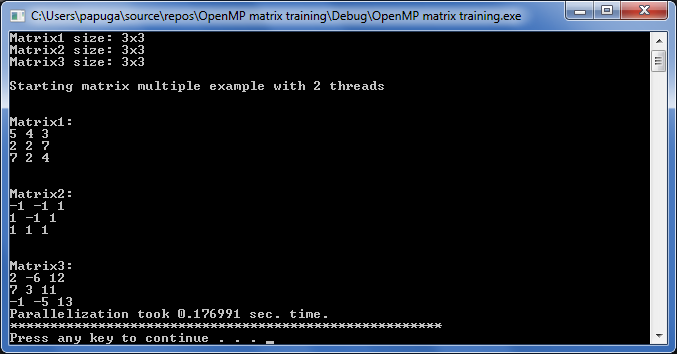
**Trumpai apie kodą:**   
Naudojamo šablono esminių „funkcijų“ nepakeičiau: išliko lygiagrečios programos zona, kurioje inicializuojama pagal reikalavimus pirmoji matrica, antroji matrica ir sudauginant gaunama trečioji matrica.

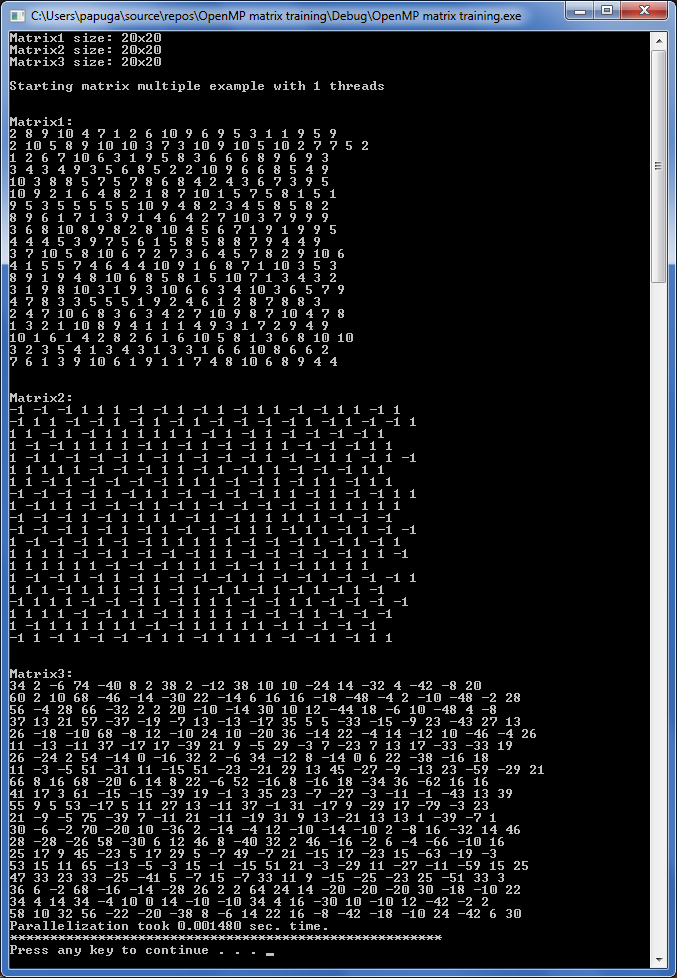
**Apie naudojamą kompiuterį:**  
Mano naudojamas kompiuteris nėra galingas, vis dėlto dviejų branduolių. Todėl patikimus rezultatus ir išvadas bus sunku pateikt įtikinamai.

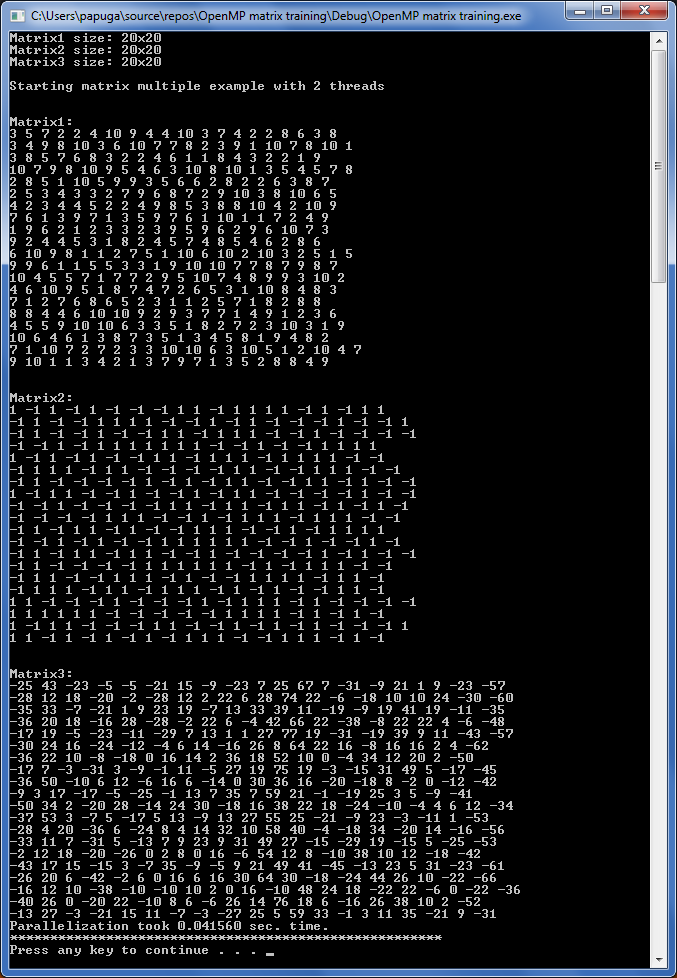
**Skaičiavimai ir rezultatai:**  
Prieš skaičiavimus esanti kompiuterio procesoriaus būsena.

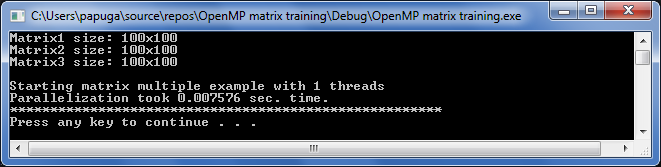
Programos (atkreipti dėmesį į matricų dydžius bei naudojamą branduolių kiekį (vienas arba du)):  
Mažos matricos:

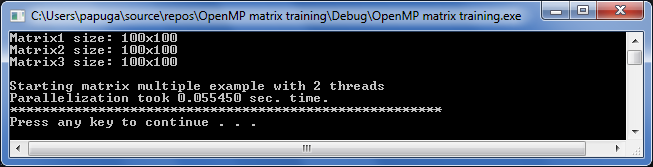




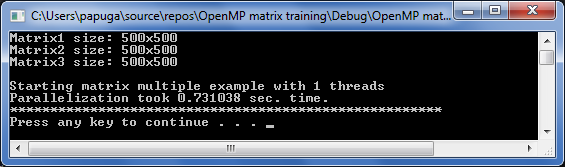


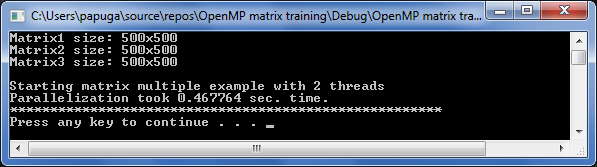


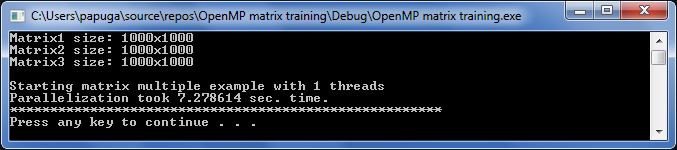




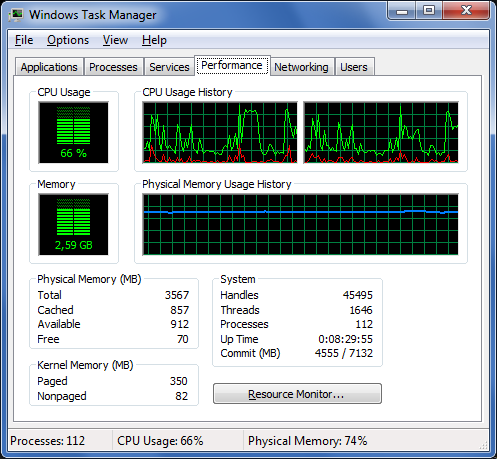
Aiškiai matoma, kad mažosiose matricose, nėra verta naudoti lygiagretūjį programavimo metodą šiam uždaviniui spręsti. Tačiau, ar didesnėse matricose yra pokyčių rezultate?

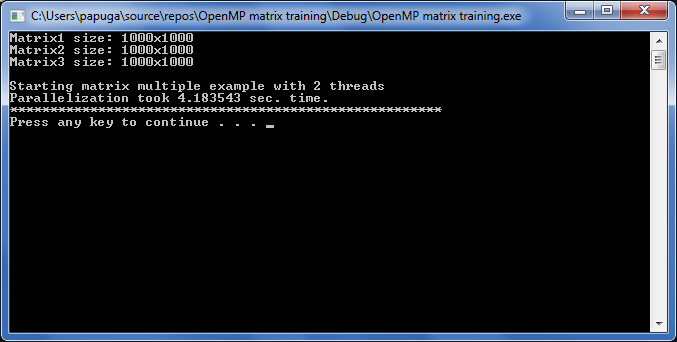
Didelės matricos:  




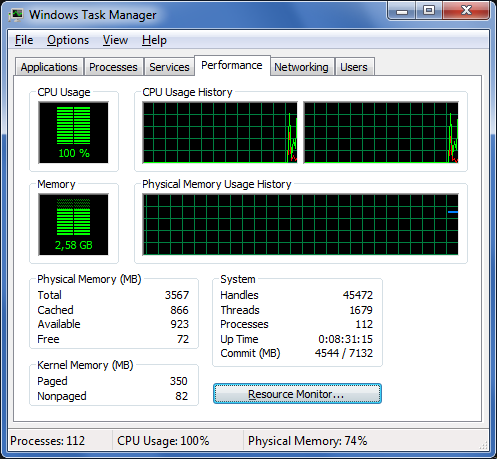


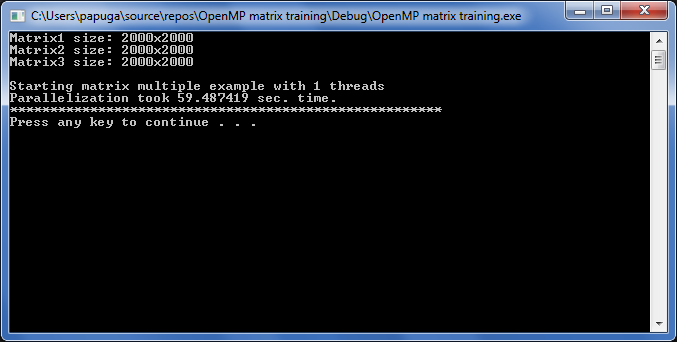
Vidutinis procesoriaus darbas, naudojant vieną branduolį:



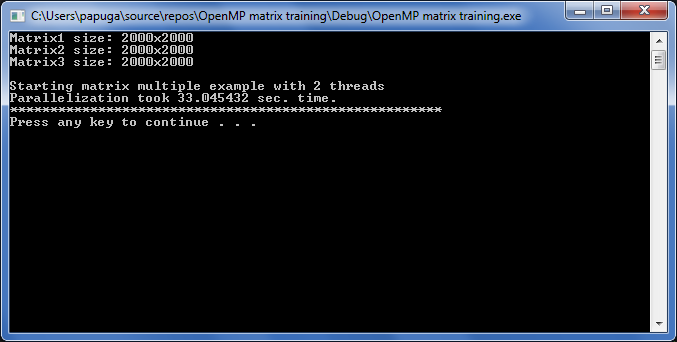


Vidutinis procesoriaus darbas, naudojant du branduolius:



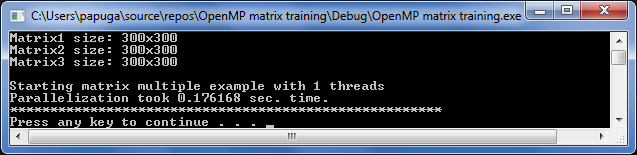
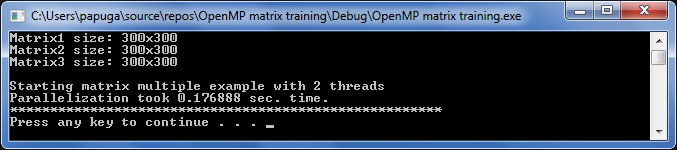


Matome, kad padidinus matricos dydį keturis kartus, skaičiavimų trukmė padidėjo apie 10 kartų (iš ~7sek. Iki ~60sec)



Naudojant du branduolius, programos efektyvumas padidėjo beveik du kartus.

Įdomu pasidarė surasti matricų dydžių ribą, kurioje išsilygina nuoseklieji skaičiavimai su lygiagrečiais. Tai turėtų būti tarp 100x100 ir 500x500, nes ties tuo lygiagretusis buvo pranašesnis.

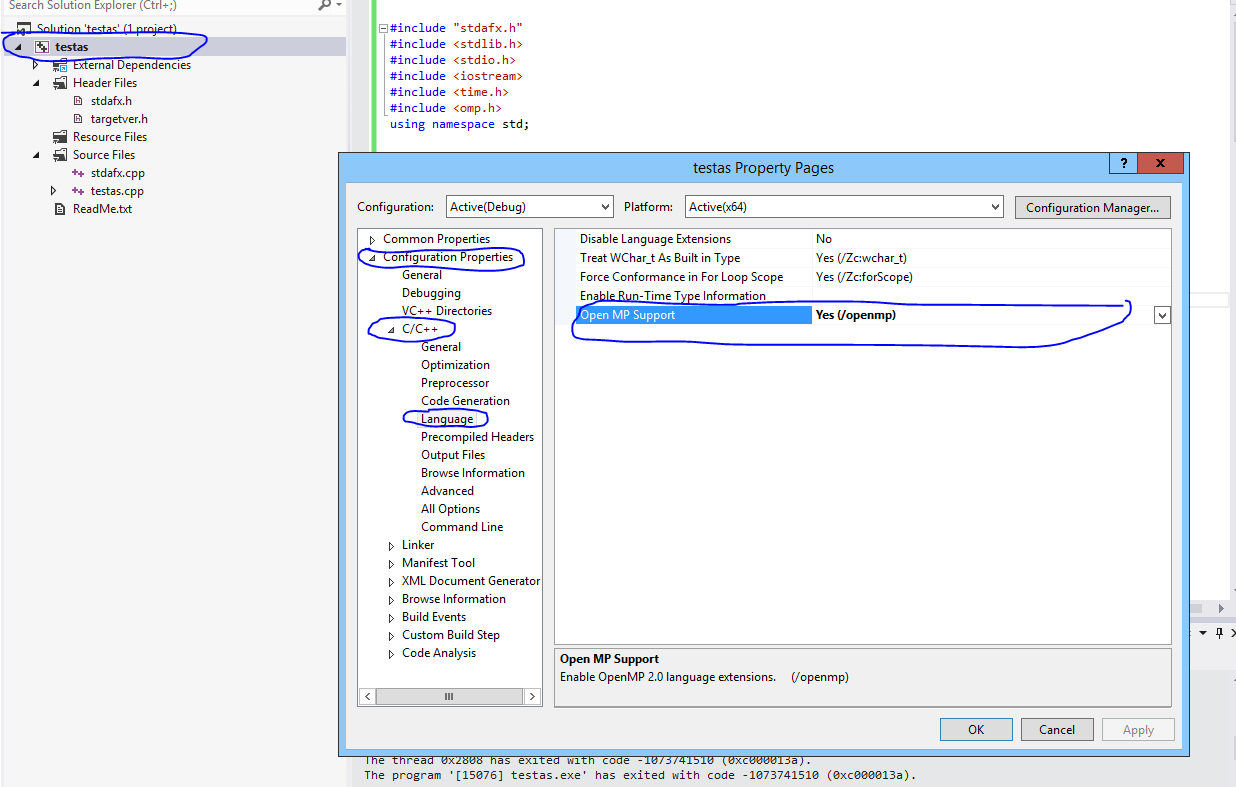
 

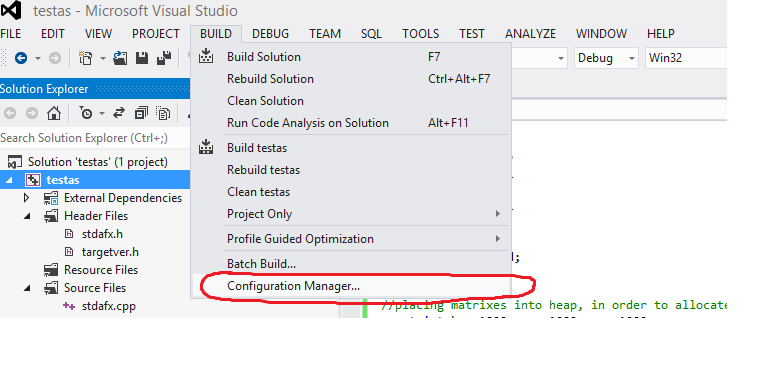
Nepaisant paklaidų, maždaug ties 300x300 matrica susilygina lygiagretusis su nuosekliuoju. Arba jei programa yra ilgesnė nei ~0.15sec, gali būti naudingiau naudoti lygiagretųjį programavimo metoda, nei nuoseklųjį.

**Išvados:**  
Iš rezultatų matoma, kad nuoseklusis metodas yra naudingas programoms, kurios darbą atlieka greičiau, nei per ~0.15sec, kitu atveju, naudingiau laiko atžvilgiu būtų naudoti lygiagretųjį. Žinoma, reikia atkreipti dėmesį, kad lygiagrečiajam reikia daugiau resursų: procesoriaus branduolių. Bent jau užimama atmintis tarp programų metodų nesikeičia.

**VARTOTOJO INSTRUKCIJA**

Projektopav -> properties



Tada BUILD -> configuration manager..

