UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET
KATEDRA ZA ASTRONOMIJU

Izveštaj o urađenom praktičnom projektu iz predmeta

Primena super-računara u astronomiji

Tema:

Paralelizacija u programskom jeziku Python

Strategije ubrzanja kod za računanje metrika za evaluaciju kvaliteta krivih sjaja AGJ u okviru LSST pregleda neba

Mentor:
Prof. dr Bojan Novaković

Student:
Isidora Jankov

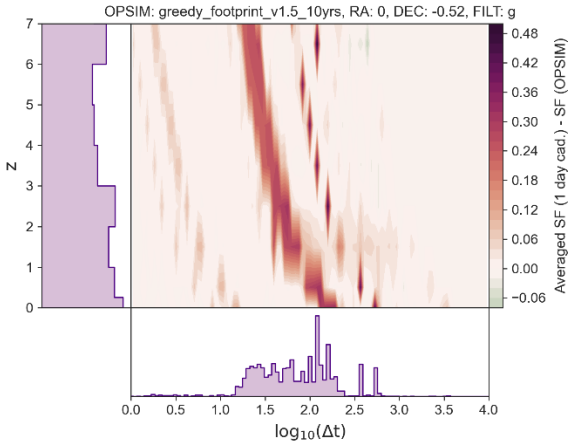
Jun, 2021. god.

Uvod

Jedan od važnih istraživačkih ciljeva Rubin Opservatorije i Legacy Survey of Space and Time (LSST) pregleda neba je istraživanje promenljivog optičkog neba. Radi se o fotometrijskom pregledu neba u trajanju od 10 godina, a prvo svetlo se očekuje krajem 2023. godine. Naš tim sa Katedre za astronomiju i Astronomske opservatorije u Beogradu učestvuje u pre-operacionim aktivnostima ovog projekta u okviru In-kind doprinosa. Jedna od aktivnosti na kojima radimo je dizajn metrika za evaluaciju kvaliteta krivih sjaja aktivnih galaktičkih jezgara. Ove metrike pružaju informacije o tome koliko su različite desetogodišnje strategije LSST posmatranja neba optimalne po pitanju kvaliteta parametara koji se mogu izvući iz krivih sjaja i iskoristiti u naučnim analizama. Pod strategijama posmatranja se podrazumevaju planovi gde se isprobavaju različiti redosledi selekcije polja, redosled izbora filtera, ekspozicija i niz drugih parametara koji su vezani za dizajn instrumenta i simulacije vremenskih uslova geografskog područja na kome je lociran teleskop tokom deset godina.

Strategije posmatranja se simuliraju pomoću aplikacije LSST OpSim, a potencijalni kandidati strategija (često ih zovemo OpSim-ovi) su dostupni za upotrebu članovima LSST naučnih kolaboracija na Sci-Serveru. Da bi se dobili vremenski trenuci posete teleskopa na datoj tački na nebu tokom deset godina pregleda neba u određenom filteru, dovoljno je pretražiti bazu podataka na Sci-Serveru uz pomoć već instaliranih LSST Python paketa.

SF metrika

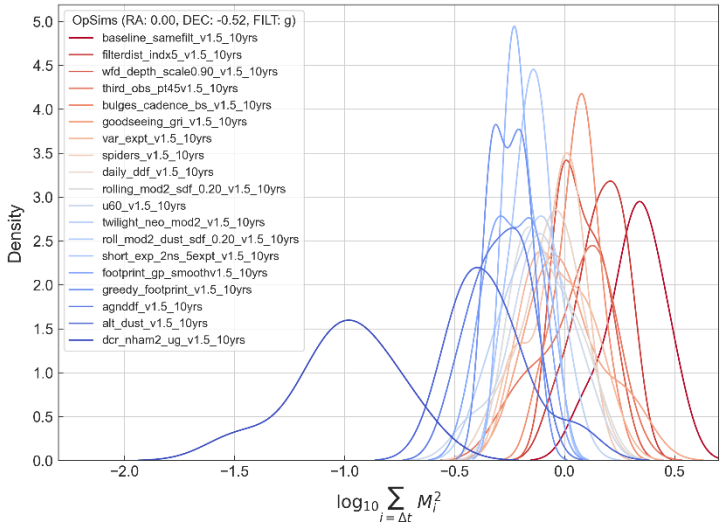


Slika 1. Mapa devijacija funkcije strukture simuliranih LSST krivih sjaja od funkcije strukture idealne simulirane krive sjaja za datu strategiju posmatranja (OpSim). Devijacije se evaluiraju po binovima vremenske skale i crvenog pomaka. Marginalni histogrami označavaju kumulativne vrednosti devijacija po osama.

Ideja iza jedne od metrika koju smo implementirali u Python-u je procena odstupanja funkcije strukture (structure function – SF) očekivanih krivih sjaja iz LSST kataloga od idealne svetlosne krive. Pod idealnom svetlosnom krivom se smatra model krive baziran na Damped Random Walk (DRW) procesu sa kadencom[1] od 1 dan (za više detalja pogledati Kovačević et al. 2021a). Sa druge strane, za LSST krive sjaja već imamo kadence za razne strategije posmatranja (one nisu uvek 1 dan), te je neophodno samo evaluirati fluks u tim trenucima pomoću modela. Nakon toga se kreiraju SF za idealne i LSST krive i računaju odstupanja. Takva metrika se računa za veštačke i LSST krive na različitim crvenim pomacima. Da bi se dobila što bolja procena, idealna kriva se dobija usrednjavanjem 50 simuliranih krivih. Odstupanja u okviru jedne OpSim strategije na određenim nebeskim koordinatama i filteru se mogu predstaviti pomoću mapa gde se na horizontalnoj osi nalaze binovi vremenske skale (poreklom od SF) a na vertikalnoj osi binovi crvenog pomaka (Slika 1). Kako bi uporedili različite OpSim strategije, ova kalkulacija se ponavlja za svaku od njih,

[1] Kadenca – vreme između dva uzastopna posmatranja određenih koordinata na nebu. Može biti ista tokom čitavog programa pregleda neba ili ne (promenljiva kadenca).

a rezultati se mogu predstaviti pomoću funkcija gustine verovatnoće ili histograma (Slika 2) sumiranih odstupanja po binovima vremenske skale ili po binovima crvenog pomaka.



Slika 2 Funkcije gustine verovatnoće devijacija funkcije strukture za različite OpSim strategije posmatranja. Na horizontalnoj osi se nalazi suma kvadrata devijacija po binovima vremenskih skala. Crvenom bojom su označene najnepogodnije strategije posmatranja, a plavom one najpogodnije (*Kovačević et al. 2021b*).

Paralelizacija koda za računanje SF metrike

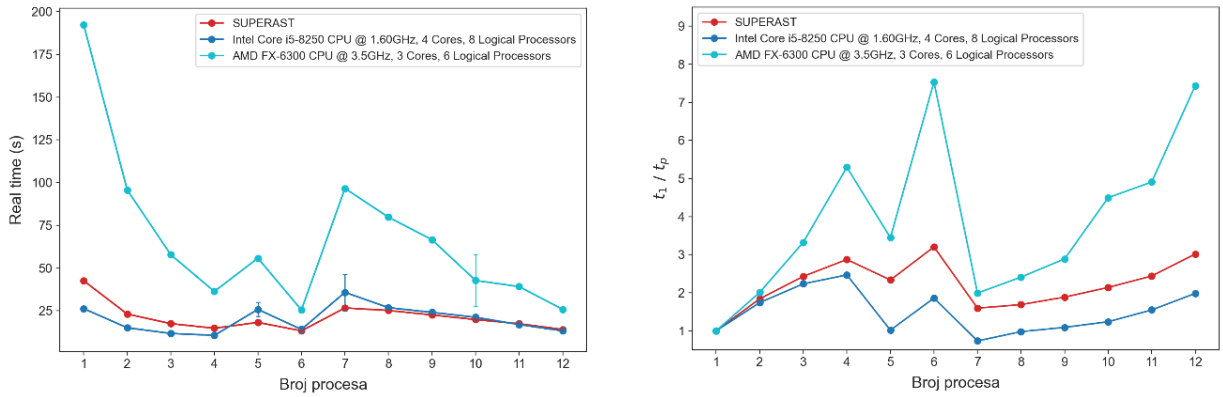
Kako se radi o kodu za koji je planirano da se testira na velikim količinama podataka, javila se potreba za njegovim ubrzanjem. Kako bi ovo postigla, koristila sam mpi4py biblioteku koja implementira MPI standard u okruženje programskog jezika Python. Opredelila sam se za paralelizaciju putem domenske dekompozicije, tj. sprovedeno je izvršavanje istih kalkulacija na različitim nezavisnim delovima ulaznih podataka. Ulazni

```
(astro) PS C:\Users\Isidora\Desktop\project> mpiexec -n 5 python para_project.py
Worker 1 received data for ['baseline_samefilt_v1.5_10yrs', 'bulges_cadence_bs_v1.5_10yrs']
Worker 1 has finished calculations
Worker 4 received data for ['goodseeing_gri_v1.5_10yrs', 'greedy_footprint_v1.5_10yrs']
Worker 4 has finished calculations
Worker 3 received data for ['filterdist_indx5_v1.5_10yrs', 'footprint_gp_smoothv1.5_10yrs']
Worker 3 has finished calculations
Worker 2 received data for ['daily_ddf_v1.5_10yrs', 'dcr_nham2_ug_v1.5_10yrs']
Worker 2 has finished calculations
Data reading finished
Worker 0 received data for ['agnddf_v1.5_10yrs', 'alt_dust_v1.5_10yrs']
Worker 0 has finished calculations
Worker 0 finished gathering the data
Finished saving first 10 opsims
Begin calculation for the remaining 2 opsims: ['roll_mod2_dust_sdf_0.20_v1.5_10yrs', 'rolling_mod2_sdf_0.20_v1.5_10yrs']

Time spent with  5  processes in seconds:
```

Slika 3 Primer pokretanja koda, ispisa statusa i rezultata.

podaci su vremenski trenuci u toku 10 godina dugog pregleda neba kada je teleskop uperio svoje oko na odabrane fiksne koordinate u g-filteru, za 12 različitih strategija posmatranja (OpSim-ova). Vremenski trenuci različitih OpSim strategija su potpuno nezavisni jedni od drugih, te je ovaj vid paralelizacije sproveden na tom nivou. Podaci iz različitih OpSim strategija su podeljeni na procese koristeći globalne MPI funkcije (scatter i gather). Kod je napisan tako da je moguće dati proizvoljan broj ulaznih OpSim strategija. Krajnji rezultat izvršavanja je sačuvana lista matrica odstupanja (poput one na Slici 1) za sve željene OpSim strategije koje se mogu kasnije iskoristiti za njihovo sveobuhvatnije poređenje (Slika 2). Primer pozivanja skripte na Windows 10 OS i prikaz rezultata je dat na Slici 3. Testiran je rad koda sa različitim brojem procesa i na različitim mašinama koristeći manje količine podataka. Rezultati su prikazani na Slici 4.



Slika 4. Rezultati testiranja ubrzanja dobijenog paralelizacijom koda. *Levo* - Ukupno vreme izvršenja programa sa različitim brojem procesa. *Desno* - Odnos vremena serijskog i paralelnog izvršenja (ubrzanje) pri pokretanju programa sa različitim brojem procesa. Kod je testiran na kućnom PC računaru, laptopu i super-računaru sa Katedre za astronomiju koji su označenim tirkiznom, plavom i crvenom linijom, respektivno.

Kao što se na desnom panelu Slike 4. može videti, PC računar (AMD FX-6300) ima najveće benefite od paralelizacije, dostižući ubrzanje od 7.5 sa 6 procesa. Međutim, on je i dalje sporiji od laptopa (Intel Core i5-8250) i SUPERASTA (Slika 4, levi panel). U režimu 1-4 procesa, laptop čak nadmašuje SUPERAST, a za veći broj procesa su skoro iste performanse. Nisam imala resursa da isprobavam veće količine podataka i broj procesa na SUPERASTU, ali očekivalo bi se da u tom režimu on svakako nadmašuje laptop. Laptop je uspeo da pokrene kod maksimalno 2.5 puta brže sa 4 procesa nego što mu je potrebno za serijsko izvršenje, dok je SUPERAST dosegao maksimum na 6 procesa sa 3.2 puta bržim izvršenjem. Sve tri mašine imaju pikove pri izvršenju sa 5 i 7 procesa, verovatno zbog toga što je test rađen sa 12 OpSim strategija, koje se dele na procese, a taj broj nije celobrojni umnožak 5 i 7, te u tom slučaju procesi nisu optimalno uposleni. Moj zaključak je da je najoptimalnije birati broj procesa tako da broj ulaznih OpSim podataka bude deljiv njime. Ako to nije moguće, onda se treba gledati da je ostatak pri deljenju što manji, jer je kod definisan tako da se preostali proračani obavljaju serijski. Takođe, nije dobro prelaziti preko broja samih jezgara u procesoru, pošto sa daljim povećavanjem procesa postoji mogućnost da vreme utrošeno na komunikaciju među njima značajno uspori kod.

U prilogu izveštaja se nalazi:

- para_proj.py – paralelizovani kod sa komentarima i dokumentacijom;
- functions.py – funkcije za simulaciju krivih sjaja AGJ i računanje SF, njih poziva paralelizovani kod;
- data – folder koji sadrži primere ulaznih podataka u g i r filteru;
- results – folder u kome program skladišti rezultate;
- Table.xlsx – tabela sa rezultatima vremenskih testova.

Reference

Kovačević, A. B. et al. 2021a, arXiv: 2105.14889 [astro-ph.GA], prihvaćen za publikaciju u MNRAS

Kovačević, A. B. et al. 2021b, arXiv: 2105.12420 [astro-ph.GA]