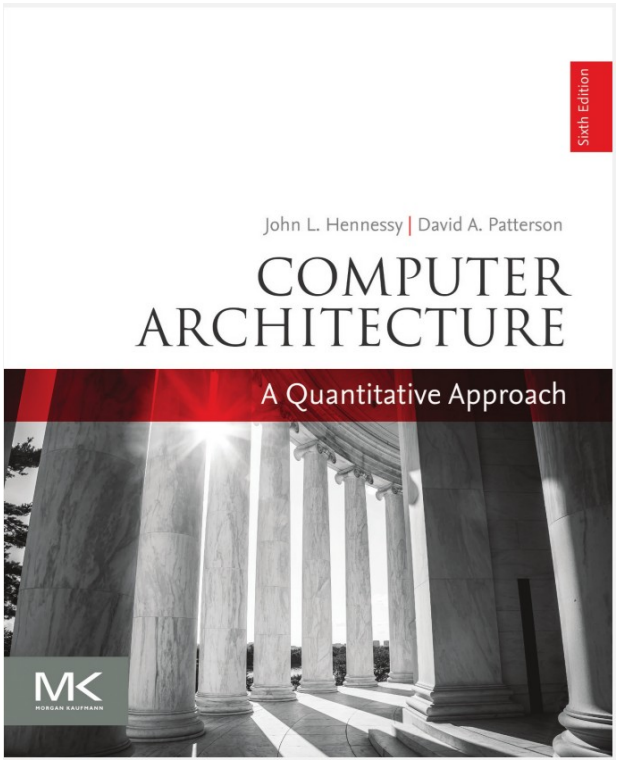
Paralelne i distribuirane arhitekture i jezici

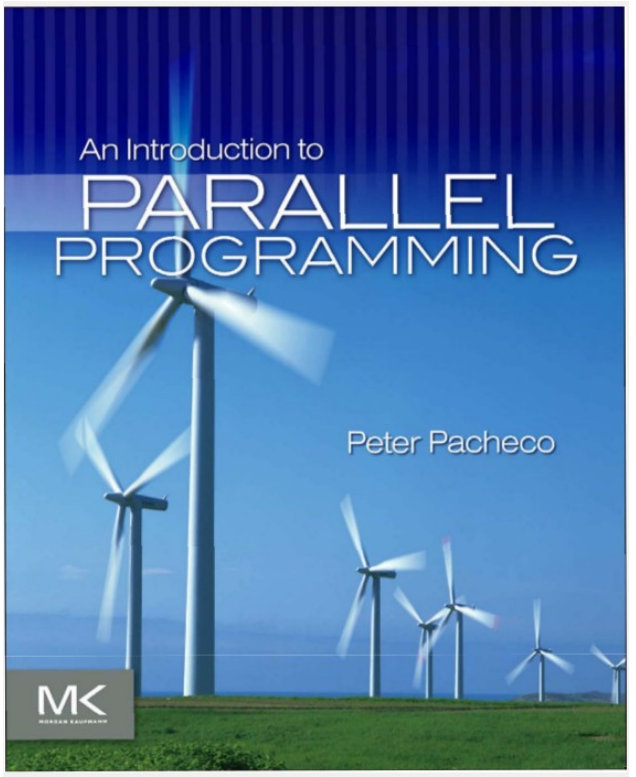
Predavanja: Žarko Živanov
Vežbe: Petar Marić

Materijal u slajdovima je najvećim delom zasnovan na knjigama “Computer Architecture: A Quantitative Approach” i “An Introduction to Parallel Programming”, navedenim u literaturi

- Razumevanje modela i koncepata savremenih paralelnih i distribuiranih računarskih arhitektura i ovladavanje tehnikama i metodama njihovog efikasnog programiranja
- Osnova za ostale predmete
- Ispit
  - Predispitne obaveze, 70 poena
    - zadatak iz multiprocessing-a, kroz projekat, 30 poena, prva polovina semestra
    - zadatak iz distribuiranih proračuna, kroz projekat, 40 poena, druga polovina semestra
  - Ispitne obaveze, 30 poena
    - usmeni

- Literatura
  - Slajdovi, dostupni kako bude išao semestar
  - Hennessy, J., Paterson, D., “Computer Architecture: A Quantitative Approach”, 6th edition, Morgan Kaufmann, 2017
  - Pacheco, P., “An Introduction to Parallel Programming”, Morgan Kaufmann, 2011

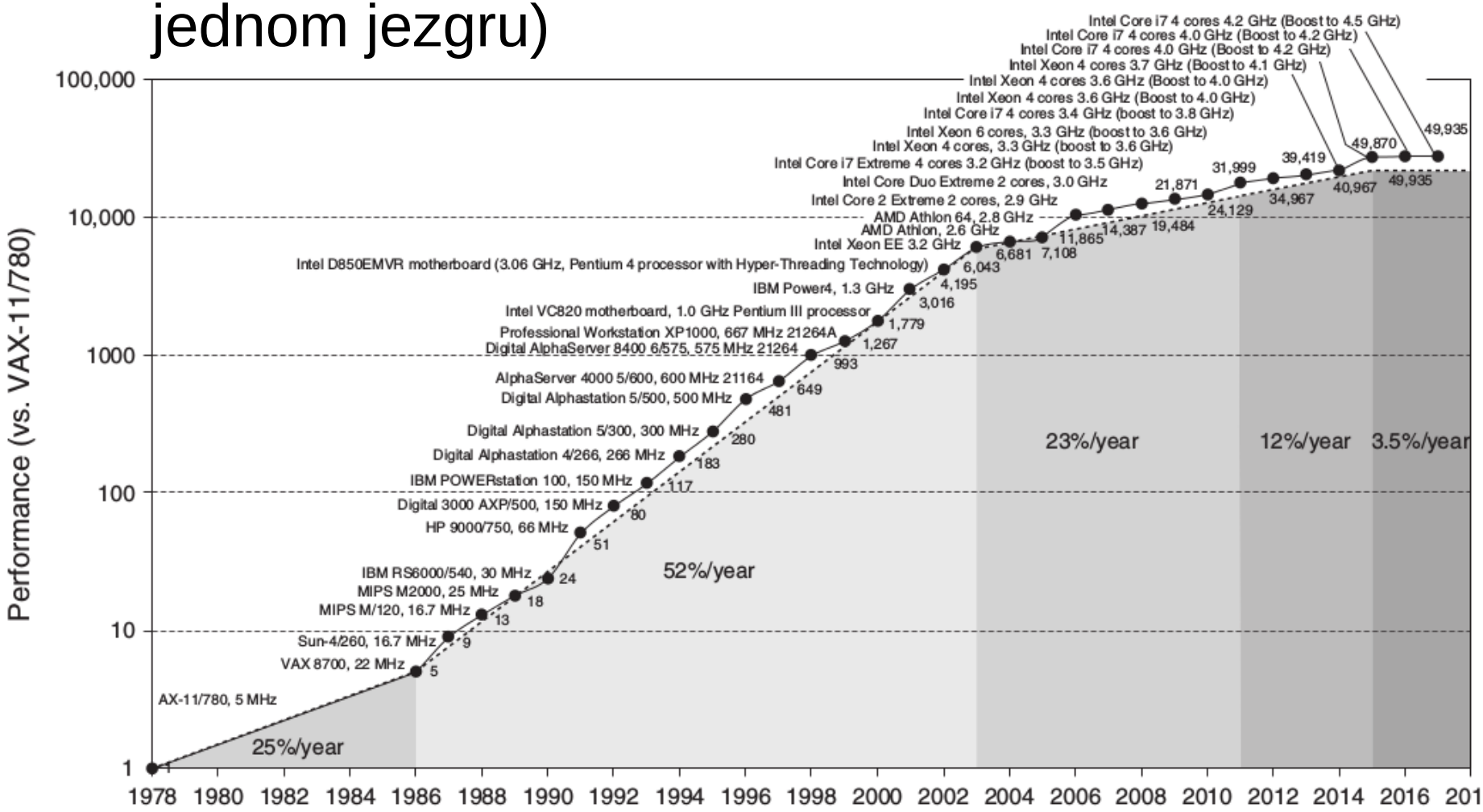




- Hardver i osnovni softver
  - 4-jezgarni procesor, ~3GHz, 32GB RAM
  - 64-bitna Linux distribucija (Ubuntu 18.04 LTS)
  - Ostalo zavisi od predmeta

- Malo istorije
  - 2003 - prvi x86 64-bitni mikroprocesor (Athlon 64)
    - Cray superkompjuteri iz 70-tih i 80-tih
    - 1992 DEC Alpha - prvi 64-bitni mikroprocesor
  - 2005 - prvi x86 mikroprocesor sa 2 jezgra (Athlon 64 X2)
    - još ranije su postojale x86 matične ploče sa dva podnožija za dva jednojezgarna mikroprocesora
  - 2006 - prvi x86 mikroprocesor sa 4 jezgra (Intel Xeon 5300)
  - Dva glavna načina ubrzanja:
    - povećanje frekvencije rada
    - povećanje gustine tranzistora / skraćenje dužine vodova

- Malo istorije
  - Od 1986 do 2002, brzina (jednojezgarnih) mikroprocesora se povećavala za oko 50% godišnje
  - Nakon 2002, to je spalo na oko 20% i manje (po jednom jezgru)



- Murov zakon (Gordon Moore, 1965)
  - Broj tranzistora po čipu se povećava duplo svake dve godine - više ne važi
- Denardovo skaliranje (Robert Dennard, 1975)
  - Gustina energije je konstantna za istu površinu na čipu; manje dimenzije -> veća gustina
  - Manja struja i napon -> veća brzina
  - Više ne važi jer se došlo do limita jačine struje i visine napona
- Amdalov zakon (G. M. Amdahl, 1967)
  - Maksimalno ubrzanje koje se može dobiti uvođenjem paralelizacije - još uvek važi!
- Umesto da se performase uduplaju svake 1.5 godine (1986-2003), danas je to na oko 20 godina...

- Malo istorije

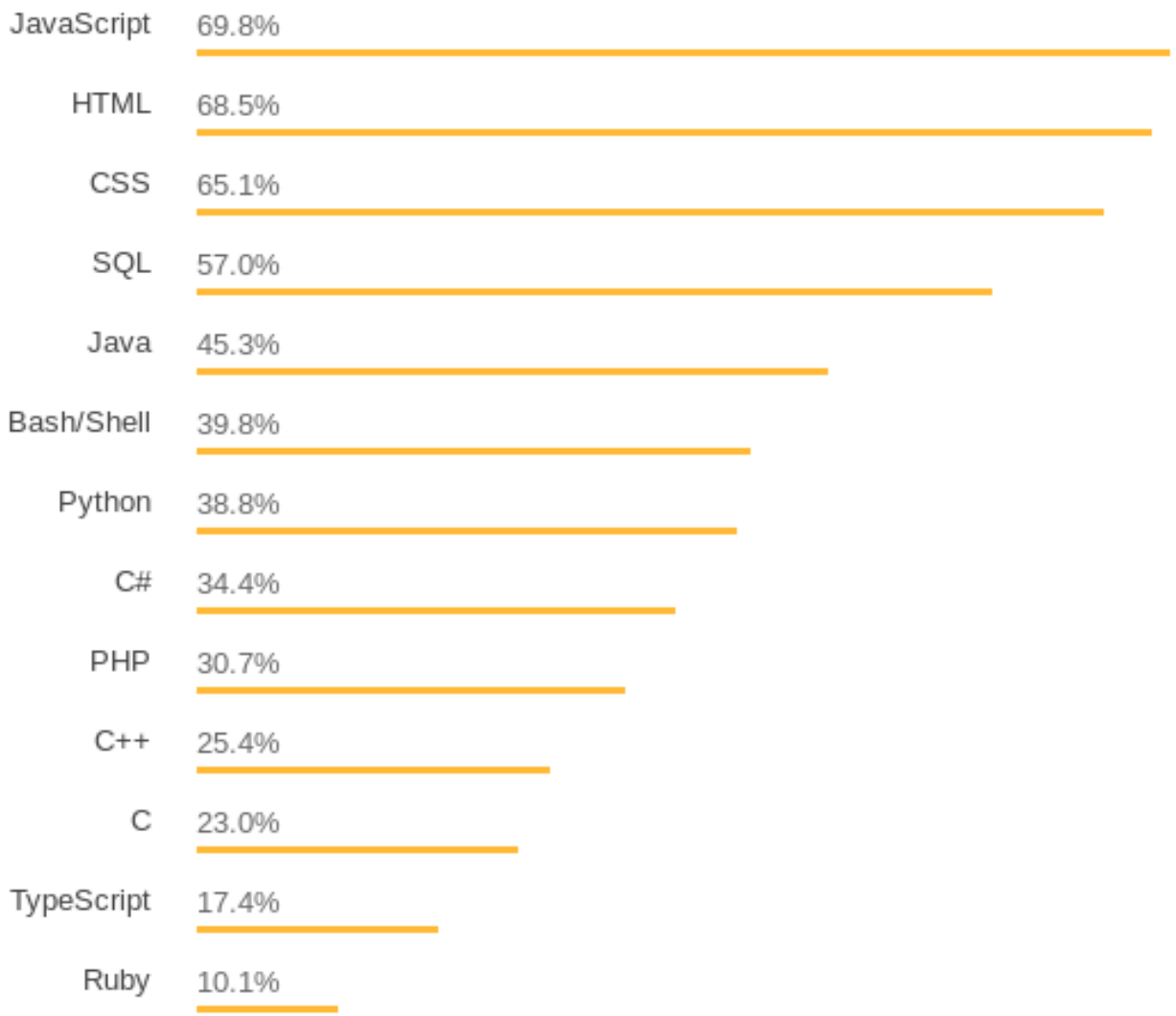
Most Popular Technologies

Programming, Scripting, and Markup Languages

All Respondents | Professional Developers

StackOverflow Survey 2018

Porast performansi računara
->
Izbor produktivnijih i manje efikasnih jezika



- Malo istorije
  - Danas
    - Telefoni sa 4-jezgarnim (mahom ARM) mikroprocesorima
    - Desktop i laptop računari standardno sa 4 jezgarnim mikroprocesorima, na raspolaganju i mnogo više (AMD EPYC sa 32 jezgra, Intel Xeon sa 20 jezgara)
    - Koje su frekvencije rada?
      - Zašto se ne ide dalje od oko 4GHz?
      - Zašto se ne ide na veoma sitne tranzistore, ~1nm?
      - Zašto se ide na više jezgara?
    - Za par stotina evra se može kupiti računar koji po performansama daleko prevazilazi računare za koje su se 80-tih morali izdvojiti milioni
    - RISC arhitektura preovladava
      - x86 procesori su interno RISC!
      - ARM je takođe RISC

- Malo istorije
  - Acorn RISC Machine
    - prvi Archimedes 400 ~3000€, 1MB, 8MHz, Risc OS

?

- Zašto paralelno programiranje?
  - Kako ubrzati izvršavanje programa?
    - nabaviti brži hardver (CPU/RAM/HDD-SSD)
    - optimizovati softver
      - korišćenje novijih kompajlera
      - pisanje koda tako da ga kompajler bolje optimizuje
      - ručno pisanje kritičnih delova koda u asembleru
    - “naterati” softver da koristi više od jednog jezgra
  - Zašto ubrzati izvršavanje programa?
    - Da li je potrebno ubrzavati sve kategorije softvera?
      - Šta je dovoljno brzo?
    - Klasični primeri
      - obrada slike i zvuka
      - (web) server
      - igre
      - naučni proračuni (fizika, biologija, meteorologija)

- Zašto paralelno programiranje?
  - Zato što je to jedini način da se iskoriste sistemi sa više procesora na rešavanju jednog problema
- Kako paralelizovati rešavanje problema?
  - kompajleri mogu prepoznati neke delove koda koji se mogu automatski paralelizovati
    - takvo deo-po-deo uključivanje paralelizma često može biti nedovoljno efikasno
  - modifikovati postojeći ili razviti novi algoritam rešavanja problema koji podržava paralelizaciju

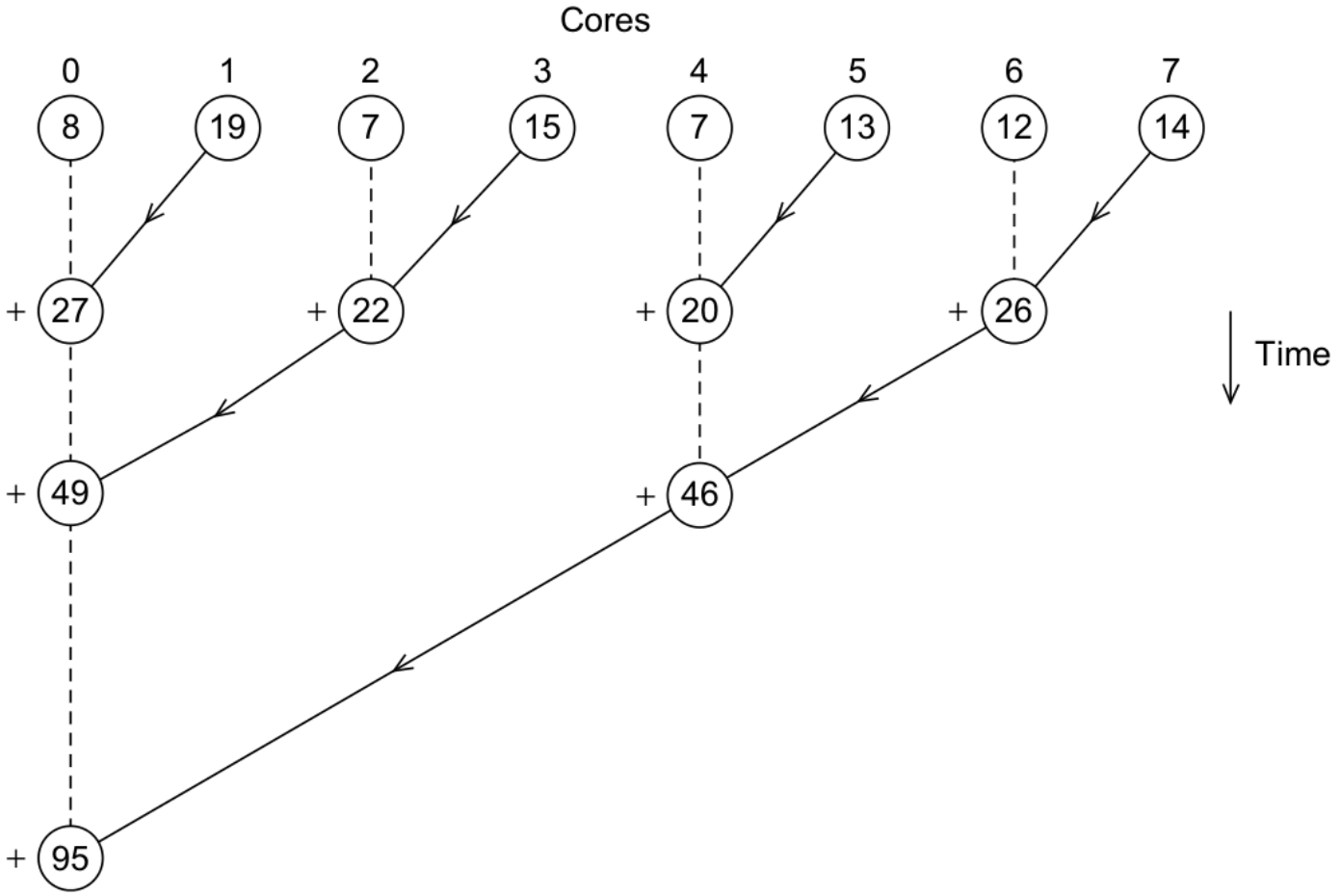
- Jednostavan primer - proračun n vrednosti i njihovo sabiranje

```
sum = 0;
  for (i = 0; i < n; i++) {
    x = Compute_next_value(. . .);
  sum += x;
}
```

```
my_sum = 0;
my_first_i = . . . ;
my_last_i = . . . ;
for (my_i = my_first_i; my_i < my_last_i; my_i++) {
  my_x = Compute_next_value(. . .);
  my_sum += my_x;
}
```

```
if (I'm the master core) {
  sum = my_sum;
  for each core other than myself {
    receive value from core;
    sum += value;
  }
} else {
  send my_sum to the master;
}
```

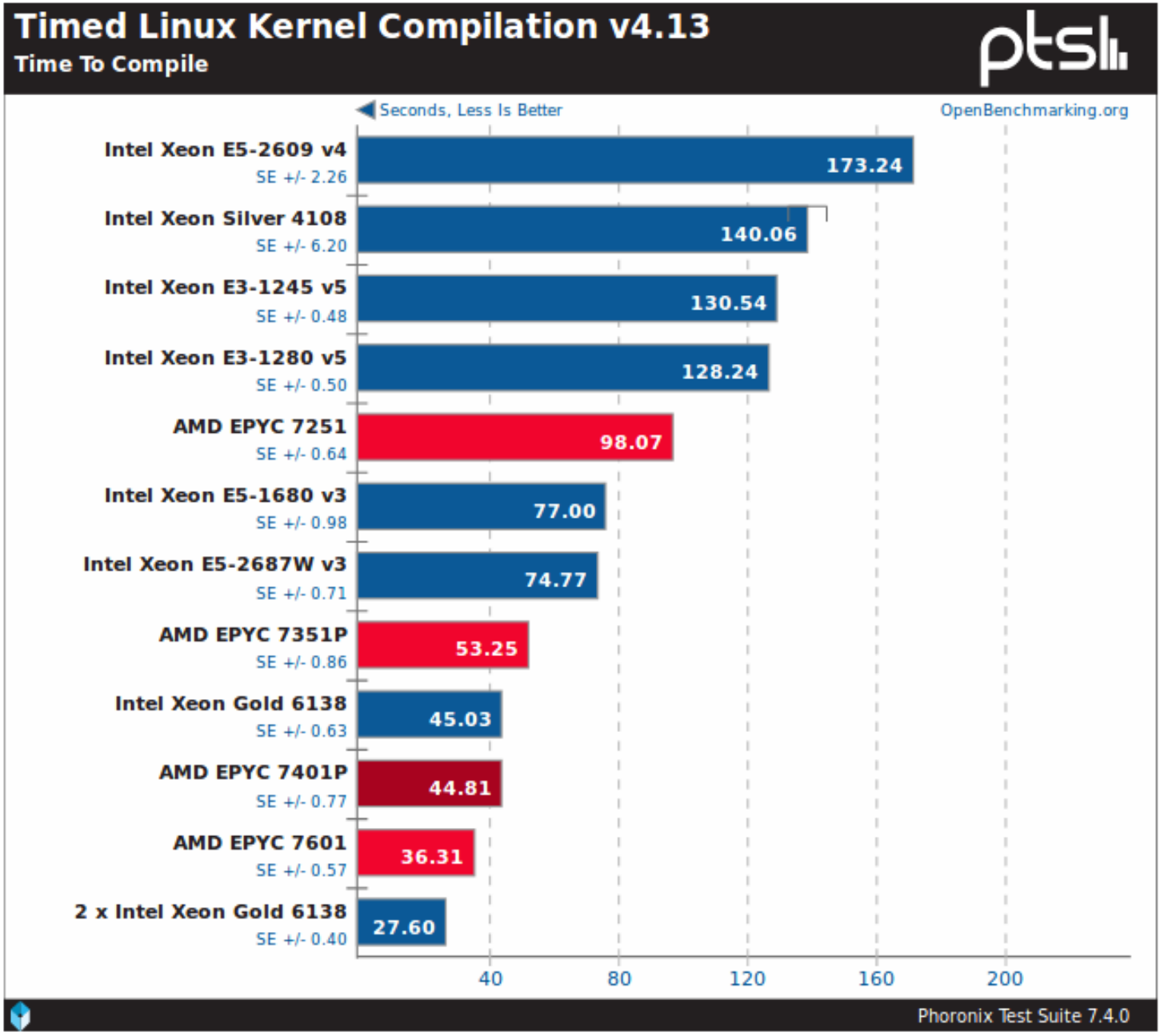
- Jednostavan primer - proračun n vrednosti i njihovo sabiranje
  - Za mali broj jezgara, ovo može da bude OK
    - koliko sabiranja treba da odradi master?
  - Šta ako imamo 1000 ili više jezgara?



- Jednostavan primer - proračun n vrednosti i njihovo sabiranje
  - Bilo bi relativno jednostavno napisati program koji bi u ovakvom primeru prepoznao da se računa suma koja se može paralelizovati
  - Šta je sa kompleksnim serijskim programima?
- Za efikasno korišćenje više jezgara, programi se od starta moraju pisati kao paralelni

- Kako pisati programe sa paralelnim izvršavanjem?
- Primer: treba oceniti 100 studentskih radova, svaki se sastoji od 5 (veoma različitih) pitanja, na raspolaganju je 5 asistenata
  - Pristup 1: svaki asistent pregleda po jedno pitanje u svih 100 radova
    - paralelizam baziran na zadatku (task-parallelism)
    - svaki asistent koristi drugačiji algoritam
  - Pristup 2: svaki asistent dobija po 20 radova i pregleda sva pitanja
    - paralelizam baziran na podacima (data-parallelism)
    - svaki asistent koristi isti algoritam

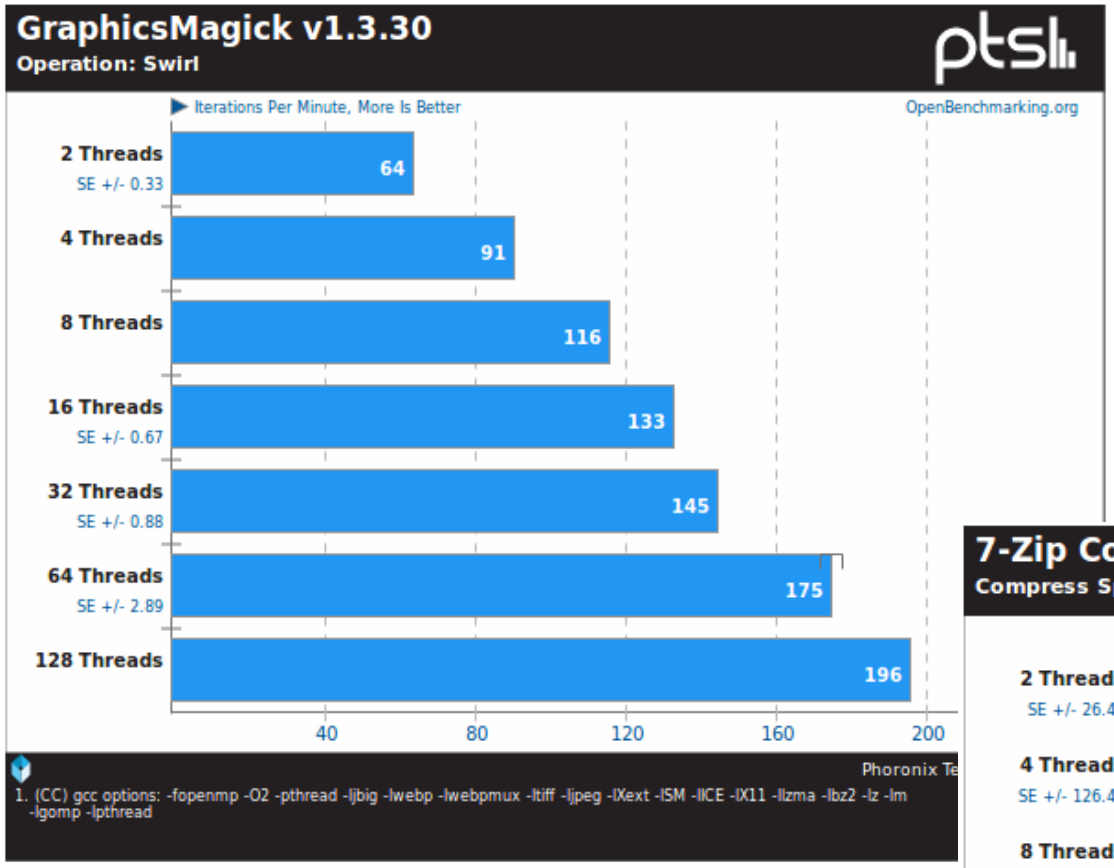
- Kompajliranje Linux kernela, v4.13
  - ~ 17 miliona linija koda
  - ~ 50 hiljada fajlova

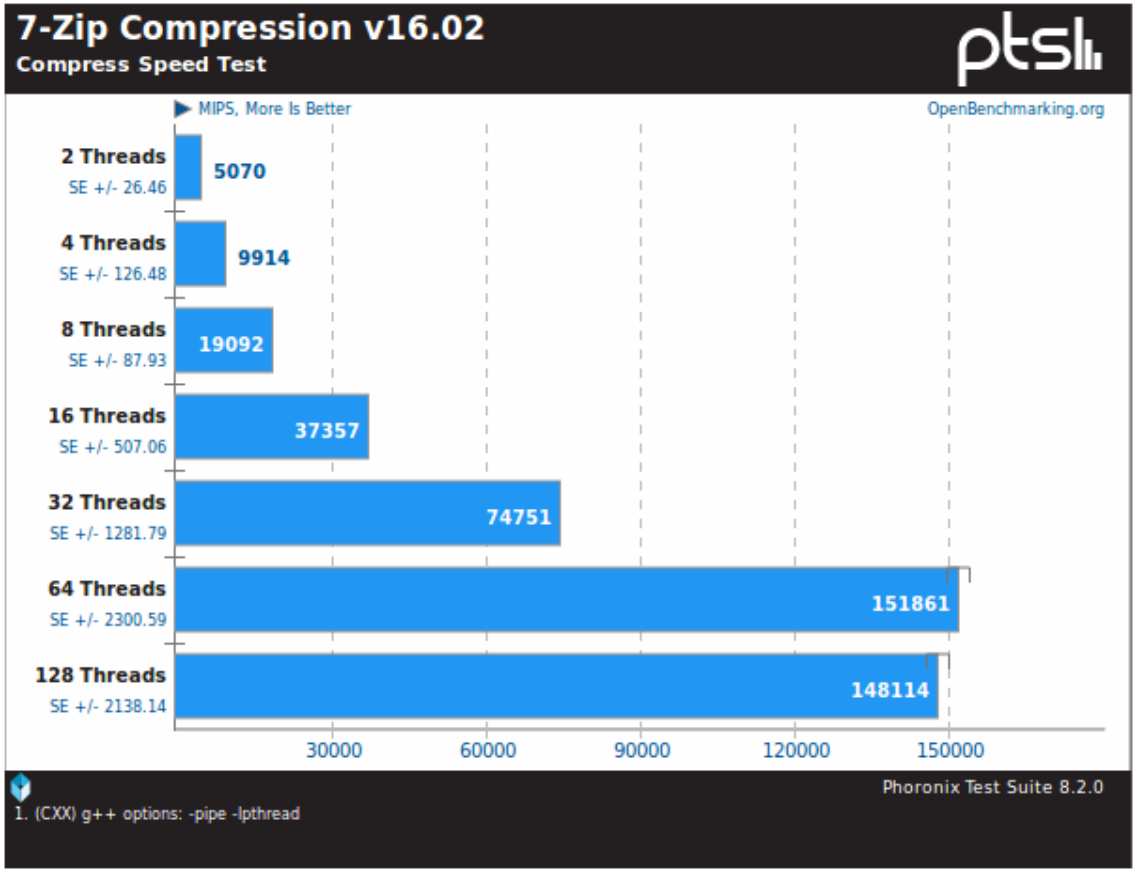


- Ako jezgra mogu da rade nezavisno, pisanje paralelnog programa je slično pisanju serijskog programa
- Zahtev za koordinacijom je ono što (umnogome) otežava stvari. Ovo je ujedno i najčešći slučaj
  - komunikacija (slanje i primanje parcijalne sume)
  - balans opterećenja (svako jezgro otprilike treba da ima istu količinu posla)
  - sinhronizacija (često nema smisla da se kreće ili nastavlja sa zadatkom dok se ne ispune neki uslovi)

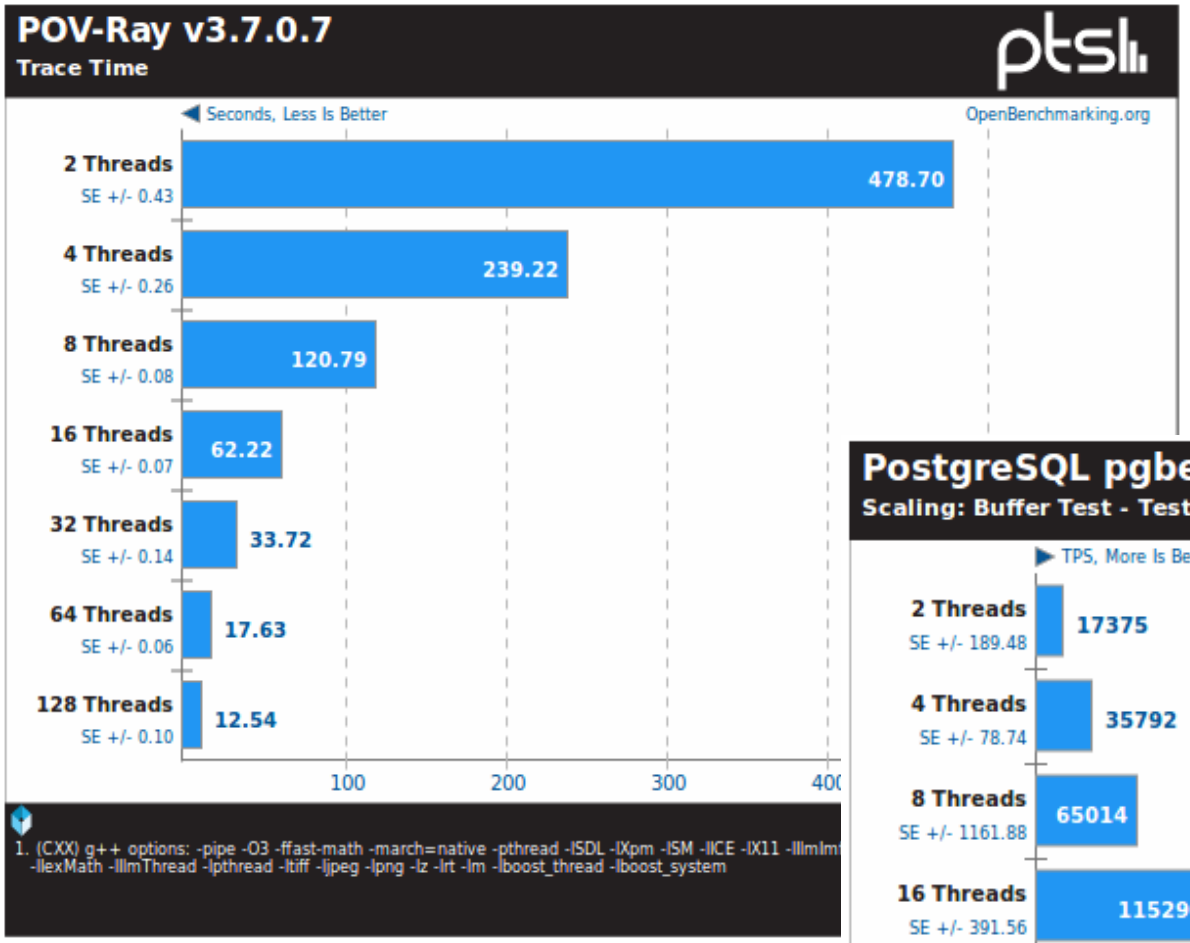
- Programski jezici sa eksplicitnim paralelizmom
  - ekstenzije C i C++ jezika
  - eksplicitno se mora napisati šta koje jezgro radi
  - često kompleksan kod za jedno jezgro
  - ali, i izuzetno brz
- Programski jezici sa ugrađenim paralelizmom
  - razvoj programa može biti lakši
  - ali najčešće na štetu efikasnosti
- Pthreads i MPI - eksplicitni paralelizam
  - Pthreads - deljena memorija (shared-memory)
  - MPI - distribuirana memorija (distributed-memory)

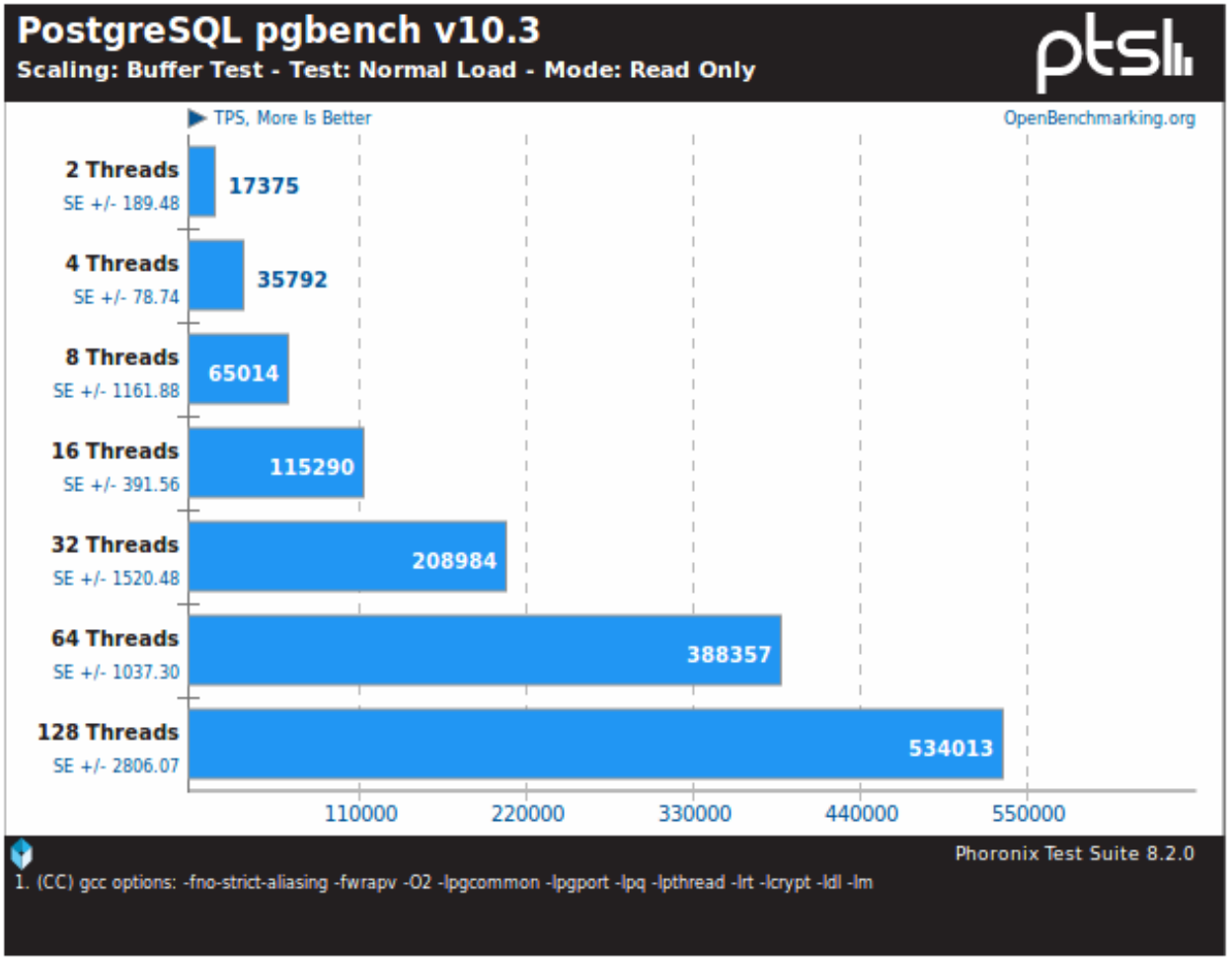
- Benchmark testovi nekih popularnih programa





- Benchmark testovi nekih popularnih programa





- Benchmark testovi nekih popularnih programa