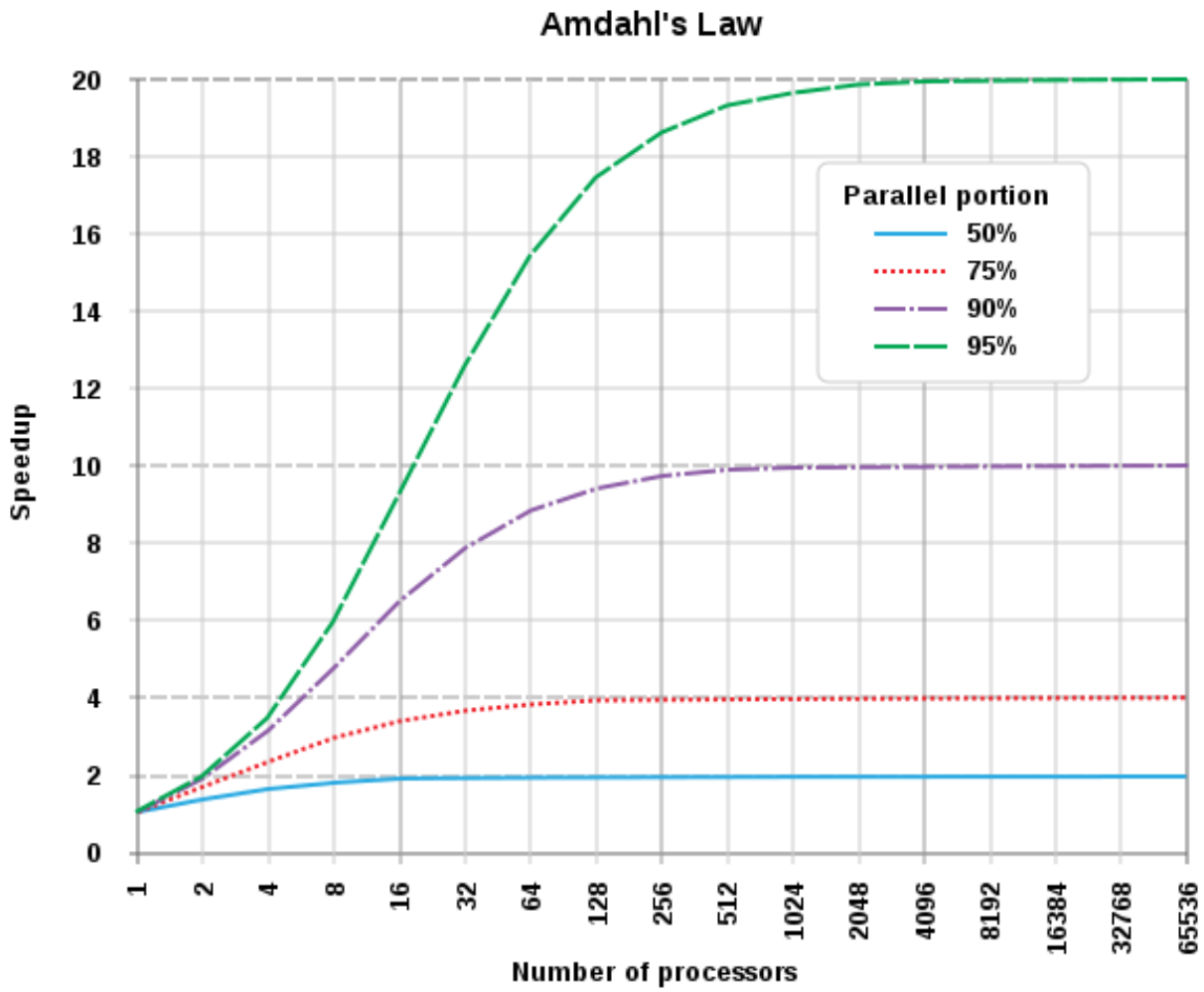
# Zakoni u paralelnom računarstvu

# Amdahlov zakon

- Dometi primene paralelizma:
  - **Amdahlov zakon** (G. M. Amdahl, 1967)
  - **Odnos serijskog i paralelnog dela obrade**
- **Paralelna obrada ima smisla** ako je:
  - **kratak sekvencijalni deo algoritma**
  - **visok stepen paralelizma**
- **Potencijalno ubrzanje programa primenom paralelizma dominatno je ograničeno delom programa koji ne može da se paralelizuje**
- Daje **gornju granicu** za ubrzanje koje se može postići paralelizacijom

Gene Amdahl
(1922–2015)

# Amdahlov zakon




Izvor: https://en.wikipedia.org/wiki/Parallel_computing

# Izvođenje Amdahlovog zakona

- Pretpostavka: najbolje moguće ubrzanje koje se može postići je **linearno**. Na osnovu prethodnog, možemo izvesti **gornju granicu za moguće ubrzanje** $S$

$$T(1) = T_{\text{ser}} + T_{\text{par}}$$

deo programa koji se **ne može efikasno paralelizovati**

deo programa koji se **može efikasno paralelizovati**

$$T(p) \geq T_{\text{ser}} + \frac{T_{\text{par}}}{p}$$

$$S(p) = \frac{T(1)}{T(p)} \leq \frac{T_{\text{ser}} + T_{\text{par}}}{T_{\text{ser}} + \frac{T_{\text{par}}}{p}}$$

# Izvođenje Amdahlovog zakona

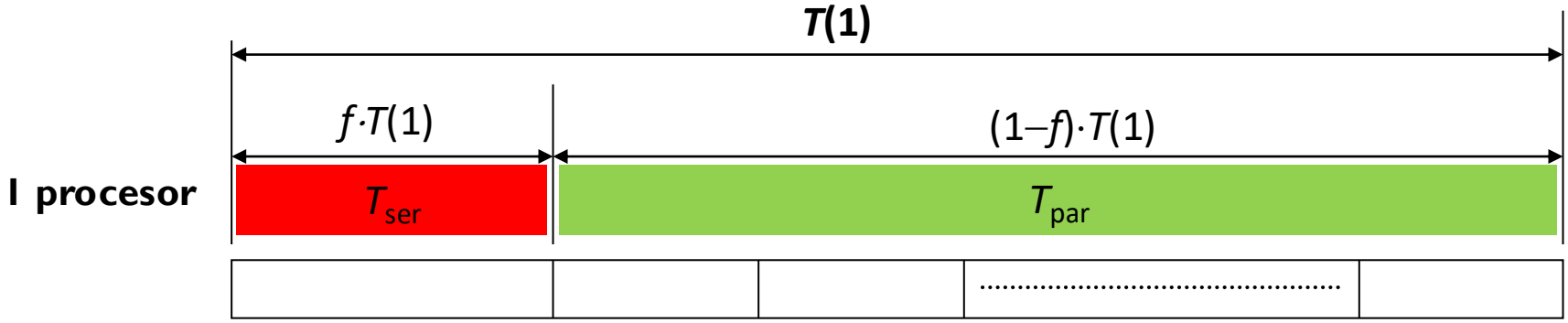
$$T_{\text{ser}} = f \cdot T(1) \text{ i } T_{\text{par}} = (1-f) \cdot T(1); \quad (0 \leq f \leq 1)$$

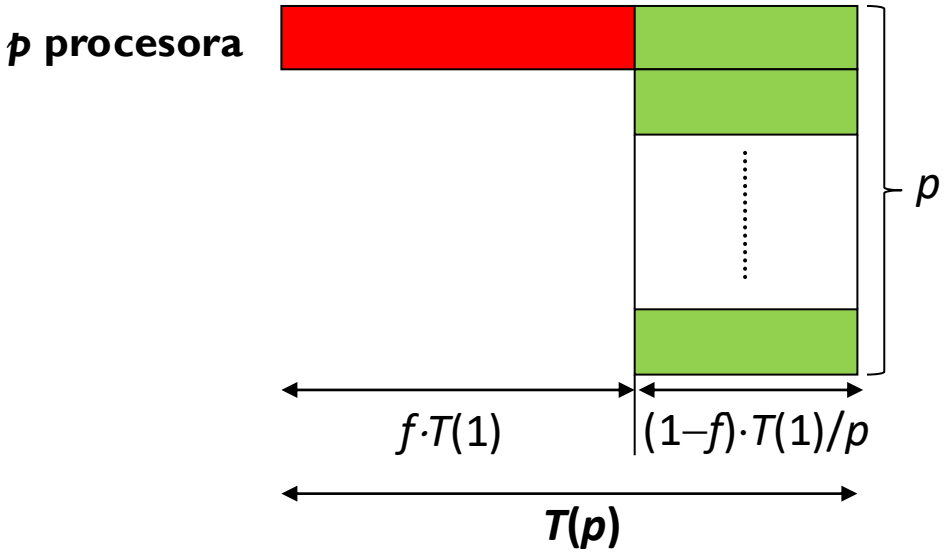
- Umesto apsolutnih vremena ($T_{\text{ser}}$ i $T_{\text{par}}$), možemo koristiti udeo, tj. **frakciju** $f$
- Zamenom u prethodno izvedenu gornju granicu, dobijamo **Amdahlov zakon**, tj. gornju granicu za ubrzanje u zavisnosti samo od $f$ i $p$

$$S(p) = \frac{T(1)}{T(p)} \leq \frac{T_{\text{ser}} + T_{\text{par}}}{T_{\text{ser}} + \frac{T_{\text{par}}}{p}} = \frac{f \cdot T(1) + (1-f) \cdot T(1)}{f \cdot T(1) + \frac{(1-f) \cdot T(1)}{p}} = \frac{1}{f + \frac{(1-f)}{p}}$$

Izvor: https://parallelprogrammingbook.org/

# Amdahlov zakon




## Amdahlov zakon

$$S(p) \leq \frac{1}{f + \frac{(1-f)}{p}}$$

Izvor: https://parallelprogrammingbook.org/

# Amdahlov zakon

$$S(p) \leq \frac{1}{f + \frac{(1-f)}{p}}$$

pri čemu $f$ označava frakciju (udeo) operacija koje se izvode **sekvencijalno** samo na jednom procesoru, dok član $(1-f)$ označava udeo operacija koje se **mogu potencijalno paralelizovati**

- Sekvencijalni deo programskog koda $f$ je broj između 0 i 1
- **Amdahlov zakon** se **u praksi** koristi **za predviđanje performansi paralelnih programa**

Izvor: https://parallelprogrammingbook.org/

# Amdahlov zakon – Primeri

1. Prilikom izvršavanja programa, 97% vremena se troši unutar petlje koja se može paralelizovati. Koje je najveće ubrzanje koje možemo očekivati od paralelne verzije programa koja se izvršava na 8 procesnih jedinica (tj. jezgara)?

$$S(8) \leq \frac{1}{0.03 + \frac{(1 - 0.03)}{8}} \approx 6.6$$

2. Ako je 20% programa inherentno sekvencijalno, koliko je maksimalno teoretsko ubrzanje paralelne verzije programa?

$$S(\infty) \leq \lim_{p \to \infty} \frac{1}{0.2 + \frac{(1 - 0.2)}{p}} = 5$$

# Skalirano ubzanje

- **Ograničenje Amdahlovog zakona**: razmatra samo situacije u kojima je veličina problema konstantna, a broj procesora promenljiv – **jaka skalabilnost** (engl. *strong scalability*)
- Kada imamo na raspolaganju više procesora, mogu se rešavati i veće instance problema – **slaba skalabilnost** (engl. *weak scalability*)
  - Onda udeo vremena $T_{\text{par}}$ može rasti brže od $T_{\text{ser}}$
- Ako se ova činjenica uključi u proračun mogućeg ubrzanja, dolazi se do pojma **skaliranog ubrzanja** (engl. *scaled speedup)*
- Može se izvesti opštiji **zakon skaliranog ubrzanja**, koji omogućava skaliranje u odnosu na složenost problema
  - **Gustafson-Barsisov zakon** je specijalni slučaj kojim se može predvideti teoretski moguće ubrzanje primenom više procesora kada **deo programa koji se može paralelizovati skalira linearno sa veličinom problema**, dok **serijski deo ostaje konstantan**

Izvor: https://parallelprogrammingbook.org/

# Izvođenje zakona skaliranog ubrzanja

$$T_{\alpha\beta}(1) = \alpha \cdot T_{\text{ser}} + \beta \cdot T_{\text{par}} = \alpha \cdot f \cdot T(1) + \beta \cdot (1 - f) \cdot T(1)$$

α - funkcija skaliranja dela programa koji **nema dobiti od paralelizacije** u odnosu na **složenost veličine problema**

β - funkcija skaliranja dela programa koji **ima dobiti od paralelizacije** u odnosu na **složenost veličine problema**

- **Skalirana gornja granica** za moguće ubrzanje:

$$S_{\alpha\beta}(p) = \frac{T_{\alpha\beta}(1)}{T_{\alpha\beta}(p)} \leq \frac{\alpha \cdot f \cdot T(1) + \beta \cdot (1 - f) \cdot T(1)}{\alpha \cdot f \cdot T(1) + \frac{\beta \cdot (1 - f) \cdot T(1)}{p}}$$

$$= \frac{\alpha \cdot f + \beta \cdot (1 - f)}{\alpha \cdot f + \frac{\beta \cdot (1 - f)}{p}}$$

Izvor: https://parallelprogrammingbook.org/

# Izvođenje Gustafson-Barsisovog zakona

Odnos skaliranja složenosti problema između delova koji se mogu i ne mogu paralelizovati → $\gamma = \frac{\beta}{\alpha}$

$$S_\gamma(p) \leq \frac{f + \gamma \cdot (1 - f)}{f + \frac{\gamma \cdot (1 - f)}{p}}$$

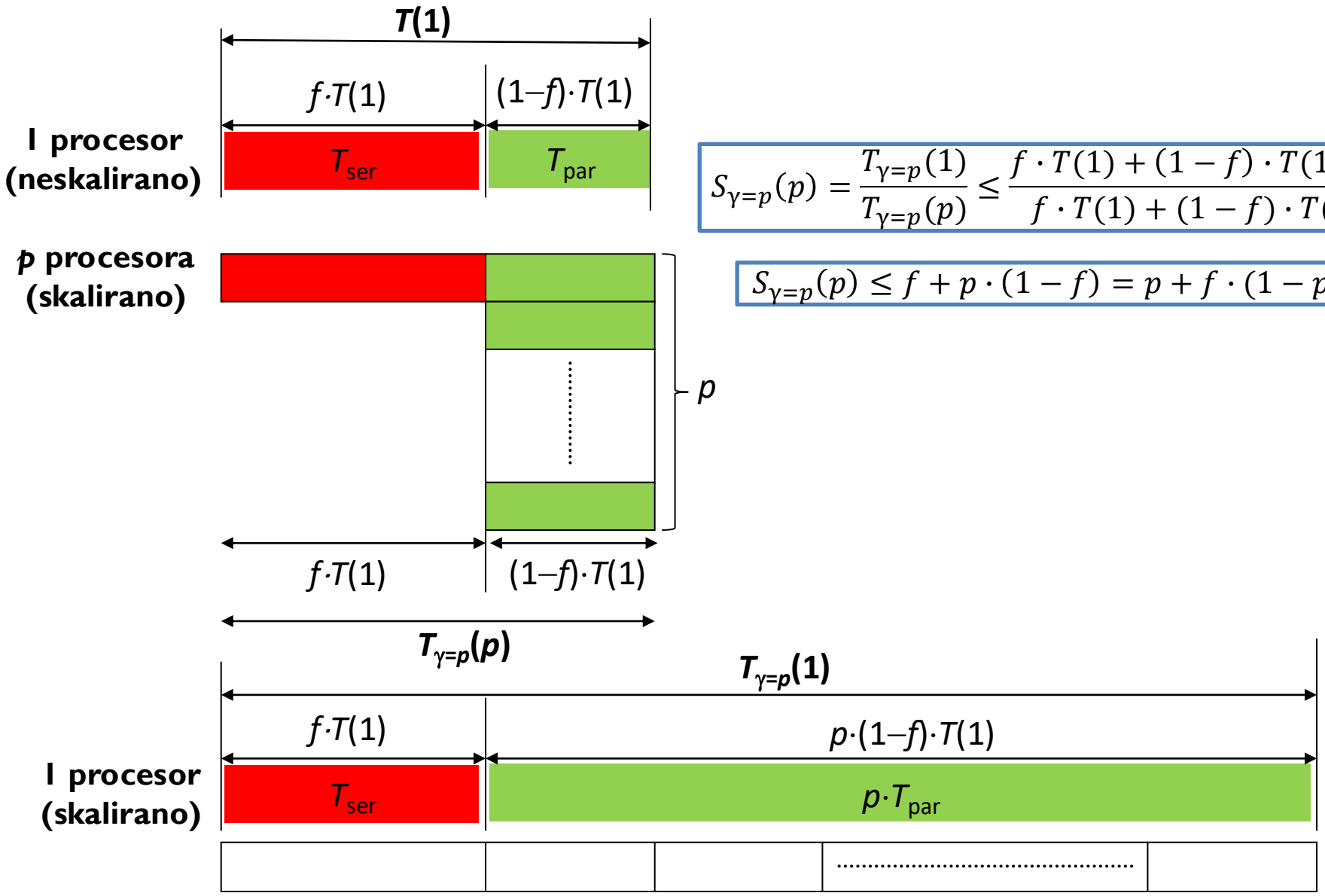
Primenom različitih funkcija za $\gamma$ dolazi se do specijalnih slučajeva:

1. $\gamma = 1$ (tj. $\alpha = \beta$): **Amdahlov zakon**
2. $\gamma = p$ (tj. $\alpha = 1; \beta = p$): deo koji se može paralelizovati raste linearno sa $p$, dok deo koji se ne može paralelizovati ostaje konstantan – **Gustafson-Barsisov zakon:**

$$S(p) \leq f + p \cdot (1 - f) = p + f \cdot (1 - p)$$

Izvor: https://parallelprogrammingbook.org/

# Gustafson-Barsisov zakon




$$S_{\gamma=p}(p) = \frac{T_{\gamma=p}(1)}{T_{\gamma=p}(p)} \leq \frac{f \cdot T(1) + (1-f) \cdot T(1) \cdot p}{f \cdot T(1) + (1-f) \cdot T(1)}$$

$$S_{\gamma=p}(p) \leq f + p \cdot (1-f) = p + f \cdot (1-p)$$

# Zakon skaliranog ubrzanja – Primer

- Paralelni program uključuje 15% sekvencijalnog posla, dok se 85%može linearno paralelizovati za datu veličinu problema. Pretpostavka je da apsolutno serijsko vreme ne raste sa skaliranjem veličine problema.

  i. Koliko ubrzanje se može postići ako koristimo 50 procesora bez skaliranja veličine problema?

$$S_{\gamma=1}(50) \leq \frac{f+\gamma\cdot(1-f)}{f+\frac{\gamma\cdot(1-f)}{p}} = \frac{1}{0.15+\frac{0.85}{50}} = 5.99$$

  ii. Ako pretpostavimo da veličina problema može da skalira naviše 100 puta, koliko ubrzanje se može postići sa 50 procesora?

$$S_{\gamma=100}(50) \leq \frac{f+\gamma\cdot(1-f)}{f+\frac{\gamma\cdot(1-f)}{p}} = \frac{0.15+100\cdot0.85}{0.15+\frac{100\cdot0.85}{50}} = 46.03$$

Izvor: https://parallelprogrammingbook.org/

# Slaba i jaka skalabilnost – Primer

- Napisati paralelni program koji postiže ubrzanje od 100 puta primenom 128 procesora.
  - i. Koji je maksimalni sekvencijalni deo programa kada ovo ubrzanje treba postići pod pretpostavkom jake skalabilnosti?

$$100 = \frac{1}{f + \frac{(1-f)}{128}} = \frac{128}{128 \cdot f + 1 - f} = \frac{128}{127 \cdot f + 1} \Rightarrow f = 0.0022$$

  - ii. Koji je maksimalni sekvencijalni deo programa kada ovo ubrzanje treba postići pod pretpostavkom slabe skalabilnosti, pri čemu $\gamma$ skalira linearno?

$$100 = 128 + f \cdot (1 - 128) = 128 - 127 \cdot f \Rightarrow f = \frac{28}{127} = 0.22$$

Izvor: https://parallelprogrammingbook.org/

# Paralelni modeli i algoritmi

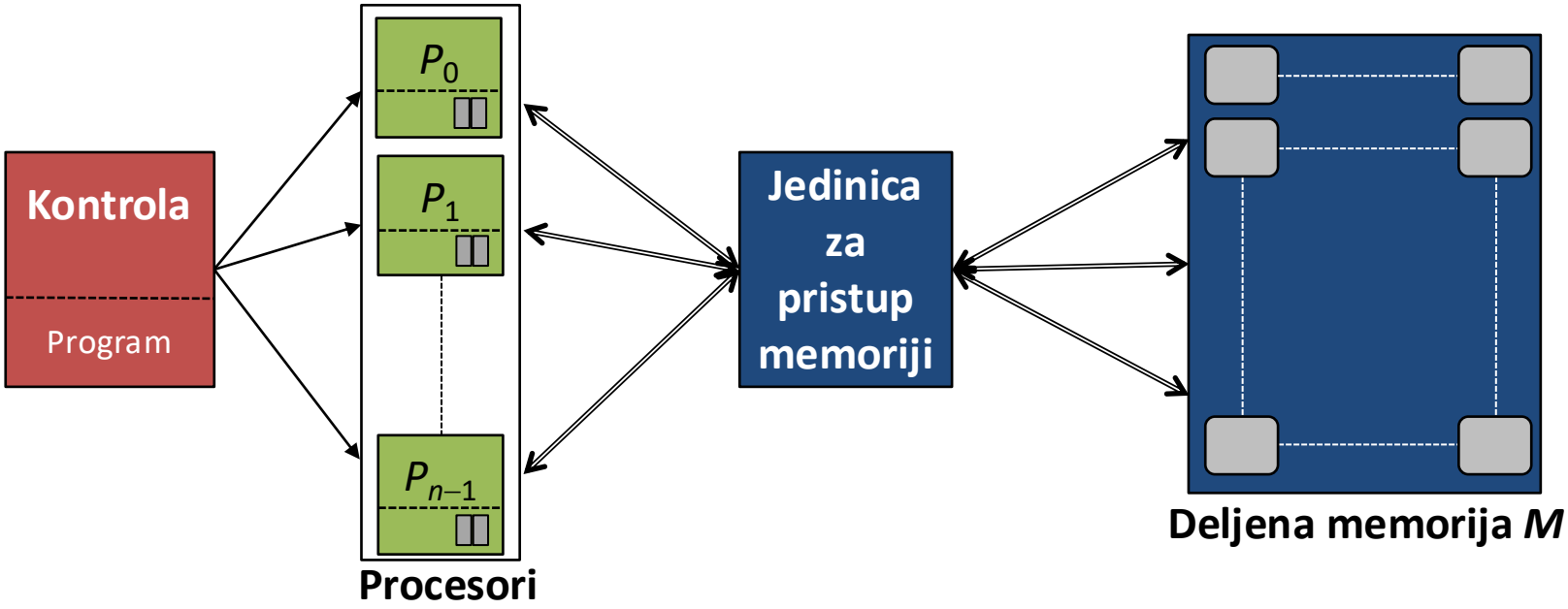
# Paralelni i distribuirani algoritmi

- **Teoretski modeli računara**
- Model **paralelnih algoritama** sa **deljenom memorijom**
  - **PRAM** (engl. *parallel random access machine*) model
- Model **paralelnih algoritama** sa **slanjem poruka**
  - Modeli su **Bulova logička kola** (engl. *Boolean circuits*) i **mreže za sortiranje** (engl. *sorting networks*)
- Model **distribuiranih algoritama** sa **slanjem poruka**
  - Model je **graf** u kome je **svaki čvor konačni automat** (engl. *finite-state machines*)

# Paralelni algoritmi sa deljenom memorijom

- **Paralelni algoritmi sa deljenom memorijom**
  - Svi procesori imaju pristup deljenoj memoriji. Projektant algoritma bira program koji izvršava svaki od procesora
  - PRAM je **idealizovana apstraktna mašina sa deljenom memorijom**, analogna RAM, **zanemaruje sinhronizaciju i komunikaciju**, cena algoritma se iskazuje kroz mere $O(vreme)$ i $O(vreme \times brojProcesora)$ i predstavlja **donju granicu** u praksi
  - Programi u deljenoj memoriji mogu da se **prošire na distribuirane** sisteme ako **operativni sistem enkapsulira komunikaciju** između čvorova i **virtuelno objedinjuje memoriju** pojedinačnih sistema
  - Model sa asinhronom deljenom memorijom je bliži realnim sistemima pošto uzima u obzir instrukcije kao što su uporedi i zameni (engl. *compare-and-swap* – CAS)

# Parallel Random Access Machine (PRAM)




- Model sa $n$ procesora $P_0,\ldots,P_{n-1}$ povezanih sa globalnom deljenom memorijom $M$
- Svakoj memorijskoj lokaciji se može uniformno pristupiti od strane bilo kog procesora u konstantnom vremenu
- Komunikacija između procesora se realizuje čitanjem i pisanjem lokacija u $M$

Izvor: https://parallelprogrammingbook.org/

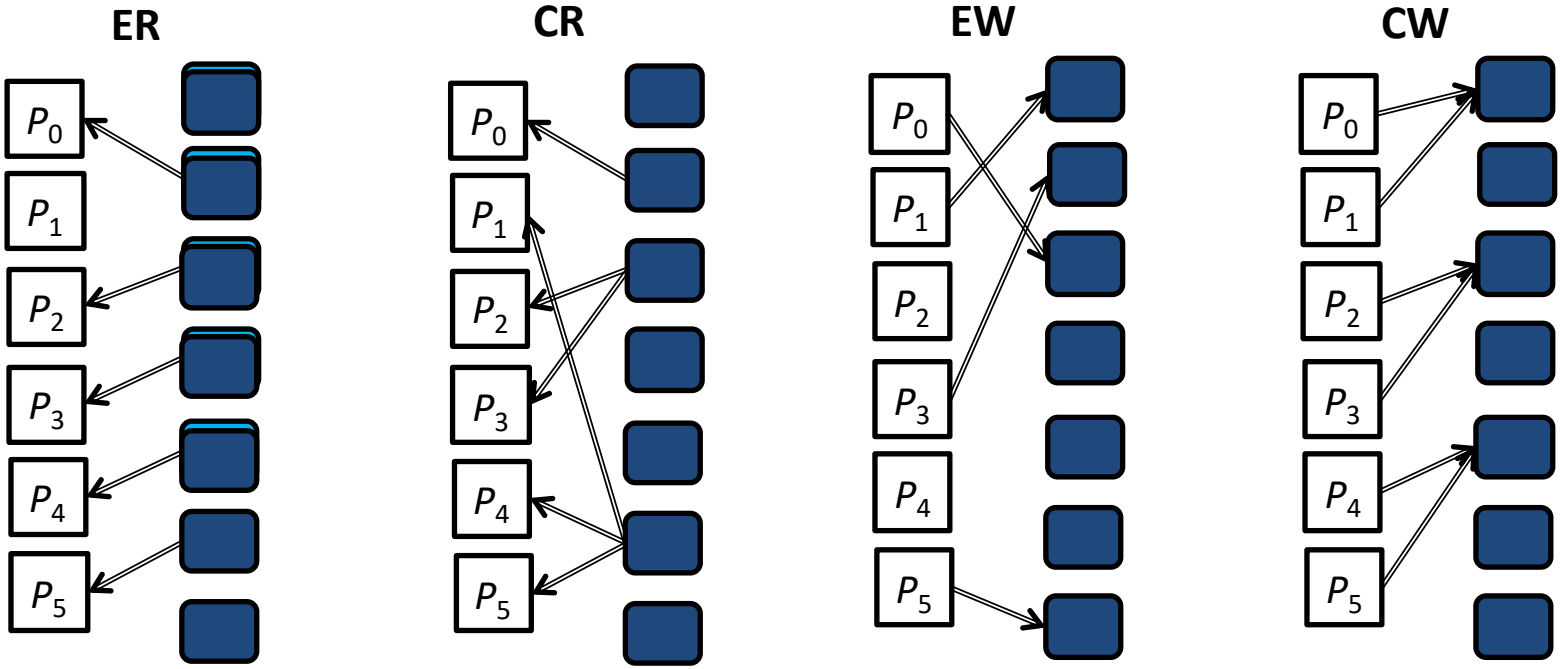
# PRAM model

- Ima $n$ identičnih procesora $P_i$, $i = 0,\ldots,n-1$ koji rade u istom koraku (engl. *lock-step*)
- U svakom koraku, svaki od procesora izvršava ciklus instrukcije u tri faze:

1. **Faza čitanja**: svaki od procesora može istovremeno da pročita jedan podatak iz (različite) lokacije deljene memorije i sačuva ga u svom lokalnom registru.
2. **Faza izračunavanja:** svaki od procesora može da izvrši neku od fundamentalnih operacija nad svojim lokalnim podacima i potom sačuva rezultat u registru.
3. **Faza upisa:** svaki od procesora može istovremeno da upiše jedan podatak u lokaciju deljene memorije, pri čemu **PRAM tip sa ekskluzivnim upisom** omogućava samo upis u različite lokacije, dok **PRAM tip sa konkurentnim upisom** omogućava da procesori upisuju na istu lokaciju (uslovi trke).

# Tipovi PRAM

- **Četiri kombinacije**, tri najpopularnije varijante EREW, CREW, CRCW
- **Exclusive Read Exclusive Write (EREW):** Samo jedan procesor sme da čita i piše u jednu deljenu memorijsku lokaciju tokom bilo kog ciklusa
- **Concurrent Read Exclusive Write (CREW):** Više procesora može čitati podatke iz iste deljene memorijske lokacije, ali upis je zabranjen




Izvor: https://parallelprogrammingbook.org/

# Tipovi PRAM

- **Concurrent Read Concurrent Write (CRCW):** dozvoljeno je istovremeno čitanje i upis u istu lokaciju u deljenoj memoriji
- Prilikom istovremenih upisa usled uslova trke neophodno je definisati koja će vrednost biti sačuvana:

1. **Prioritetni CW:** procesori imaju različiti prioritet, onaj sa najvišim uspeva da upiše vrednost
2. **Proizvoljni CW:** slučajno izabrani procesor upisuje vrednost
3. **Zajednički CW:** ako su sve vrednosti jednake, onda se vrši upis, u suprotnom lokacija ostaje nepromenjena
4. **Kombinovani CW:** sve vrednosti koje treba upisati se kombinuju u jedinstvenu vrednost primenom asocijativnih binarnih operacija (npr. sabiranje, množenje, maksimum/minimum, logičko I/ILI, …)

Izvor: https://parallelprogrammingbook.org/

# Složenost paralelnih algoritama

- Ako problem odlučivanja može da se reši u **polilogaritamskom vremenu** primenom **polinomnog broja procesora** kaže se da je u **klasi NC** ("*Nick's Class*" – Stephen Cook dao ime po Niku Pipengeru)
- U teoriji kompleksnosti, klasa **NC** je **skup problema odlučivanja** koji su odlučivi u **polilogaritamskom vremenu** na **paralelnom računaru** sa **polinomnim brojem procesora**. Problem je u klasi **NC** ako postoje konstante $c$ i $k$ takve da problem može da se reši u vremenu $O(\log^c n)$ primenom $O(n^k)$ paralelnih procesora
- Klasa NC može se jednako dobro definisati primenom formalizama PRAM-a ili Bulovih kola, u suštini predstavlja **klasu problema efikasno rešivih na paralelnim računarima**
- Nerešen problem: da li je P jednako NC? Verujemo da P nije jednako NC, ali ne možemo to i da dokažemo