07B – Implementace CA pomocí Python+MPI

Biologií inspirované počítače

Radim Kubiš

xkubis03@stud.fit.vutbr.cz

Vysoké učení technické v Brně
Fakulta informačních technologií

10. května 2014

Zadání projektu

1. Prostudovat možnosti knihovny *mpi4py*
2. Implementovat pomocí knihovny *mpi4py* paralelní výpočet
   - Conwayovy hry Life
   - Alespoň jednoho vícestatového CA
3. Provést sadu experimentů
   - Různá nastavení velikosti CA
   - Různé stupně paralelizace výpočtu
4. Zhodnotit urychlení vůči sekvenčnímu výpočtu v Pythonu
5. Posoudit
   - Výhody
   - Nevýhody
   - Použitelnost

Software a hardware

Použitý software a knihovny

- Python 3.4.0 (`http://www.python.org/`)
- NumPy 1.8.1 (`http://www.numpy.org/`)
- MPI4Py 1.3.1 (`http://mpi4py.scipy.org/`)

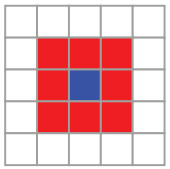
Testovací sestava

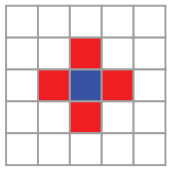
- 2× Intel® Xeon® E5430
  - Počet jader: 4 (8 celkem)
  - Frekvence: 2,66 GHz
- 16 GB DDR3 1333 MHz
- Fedora 18 (Spherical Cow, 64 bitů)

Implementované CA

Conwayova hra Life

- 2 stavy, 4 pravidla, Moorovo okolí


Moorovo okolí

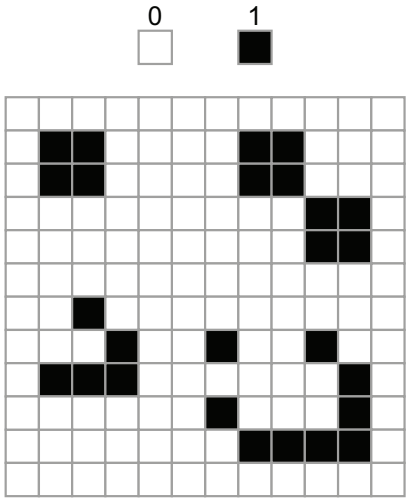

von Neumannovo okolí

Langtonova smyčka

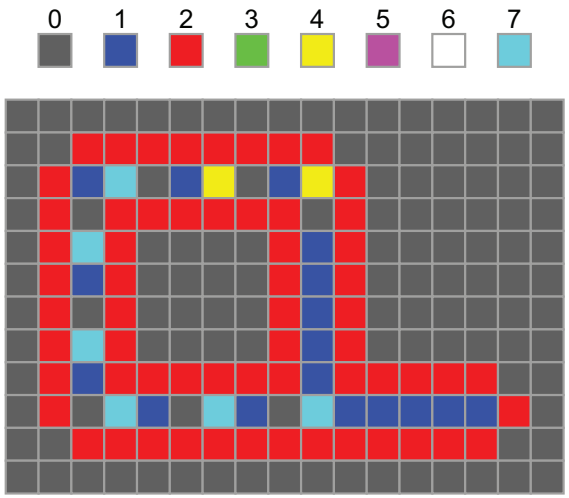
- 8 stavů, 219 pravidel (+ rotace), von Neumannovo okolí

Bylova smyčka

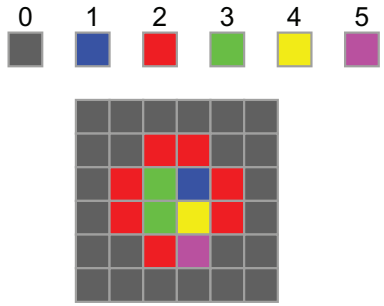
- 6 stavů, 145 pravidel (+ rotace), von Neumannovo okolí



Conwayova hra Life



Langtonova smyčka



Bylova smyčka

Metody výpočtu nových stavů

Ošetření hranic plochy CA

- Conwayova hra Life: propojení hranic
- Langtonova a Bylova smyčka: statický stav

Způsoby rozdělení plochy mezi procesory

- Sekvenční zpracování
- Zpracování po jedné buňce
- Zpracování po jednom řádku
- Zpracování po bloku rozptýlených řádků
- Zpracování po souvislých blocích řádků

Paralelní zpracování využívá vždy 1 procesor jako řídící a ostatní procesory jako výpočetní.

Sekvenční zpracování

Algoritmus výpočtu

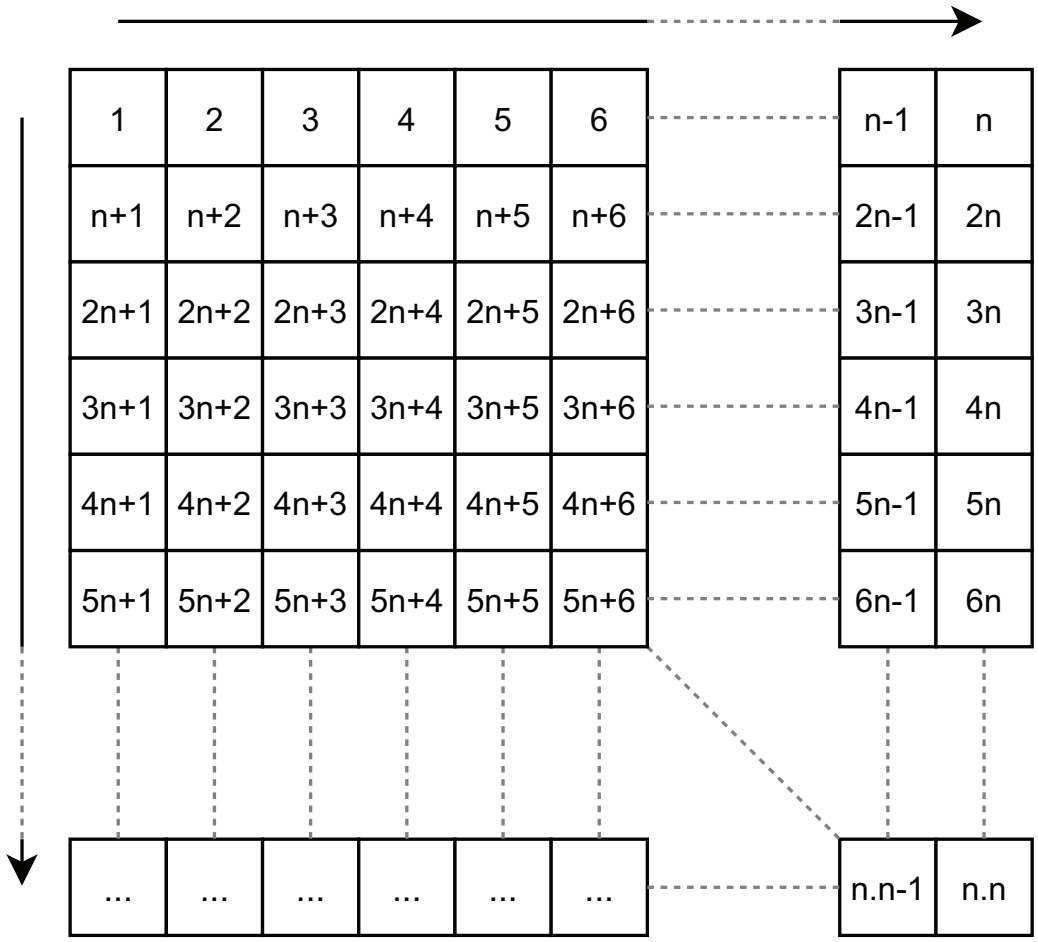
for s=1 **to** STEPS:

  for i=0 **to** N-1:

    for j=0 **to** N-1:

      CA2[i, j] = newVal(i, j)

  CA1, CA2 = CA2, CA1



Sekvenční zpracování

Zpracování po jedné buňce

Algoritmus řízení

```
for s=1 to STEPS:
    broadcastSend(CA1)
    for i=0 to (N·N)-1:
        CA2[i/N][i%N] = recv(newCell)
    CA1, CA2 = CA2, CA1
```

Algoritmus výpočtu

```
for s=1 to STEPS:
    myCA = broadcastReceive(CA1)
    i = rank-1
    while i < (N·N):
        newCell = newVal(i/N, i%N)
        send(newCell)
        i += shift
```

1	1	1	2	2	2	3	3
3	4	4	4	5	5	5	6
6	6	7	7	7	8	8	8
9	9	9	10	10	10	11	11
11	12	12	12	13	13	13	14
14	14	15	15	15	16	16	16
17	17	17	18	18	18	19	19
19	20	20	20	21	21	21	22

Zpracování po jedné buňce

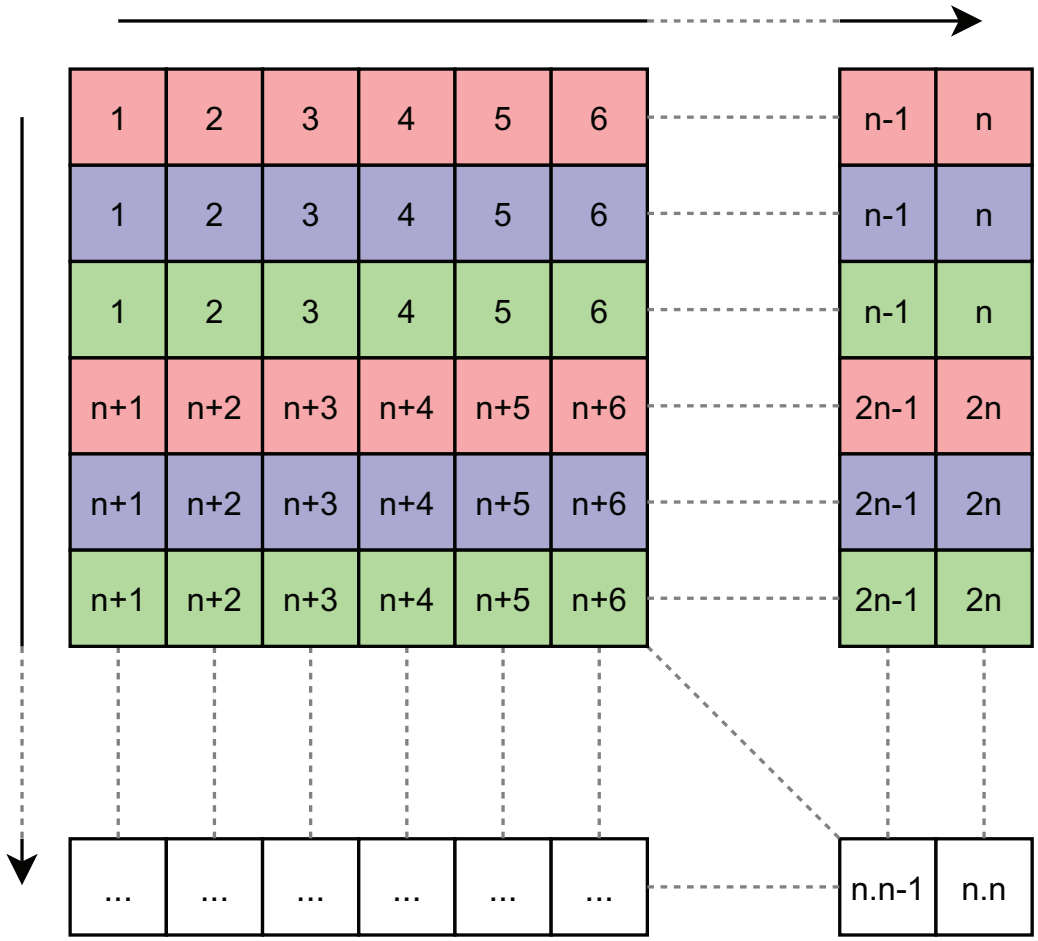
Zpracování po jednom řádku

Algoritmus řízení

```
for s=1 to STEPS:
    broadcastSend(CA1)
    for i=0 to N-1:
        CA2[i] = receive(newRow)
    CA1, CA2 = CA2, CA1
```

Algoritmus výpočtu

```
for s=1 to STEPS:
    myCA = broadcastReceive(CA1)
    i = rank-1
    while i < N:
        newRow = []
        for j=0 to N-1:
            newRow.add(newVal(i, j))
        send(newRow)
        i += shift
```



Zpracování po jednom řádku

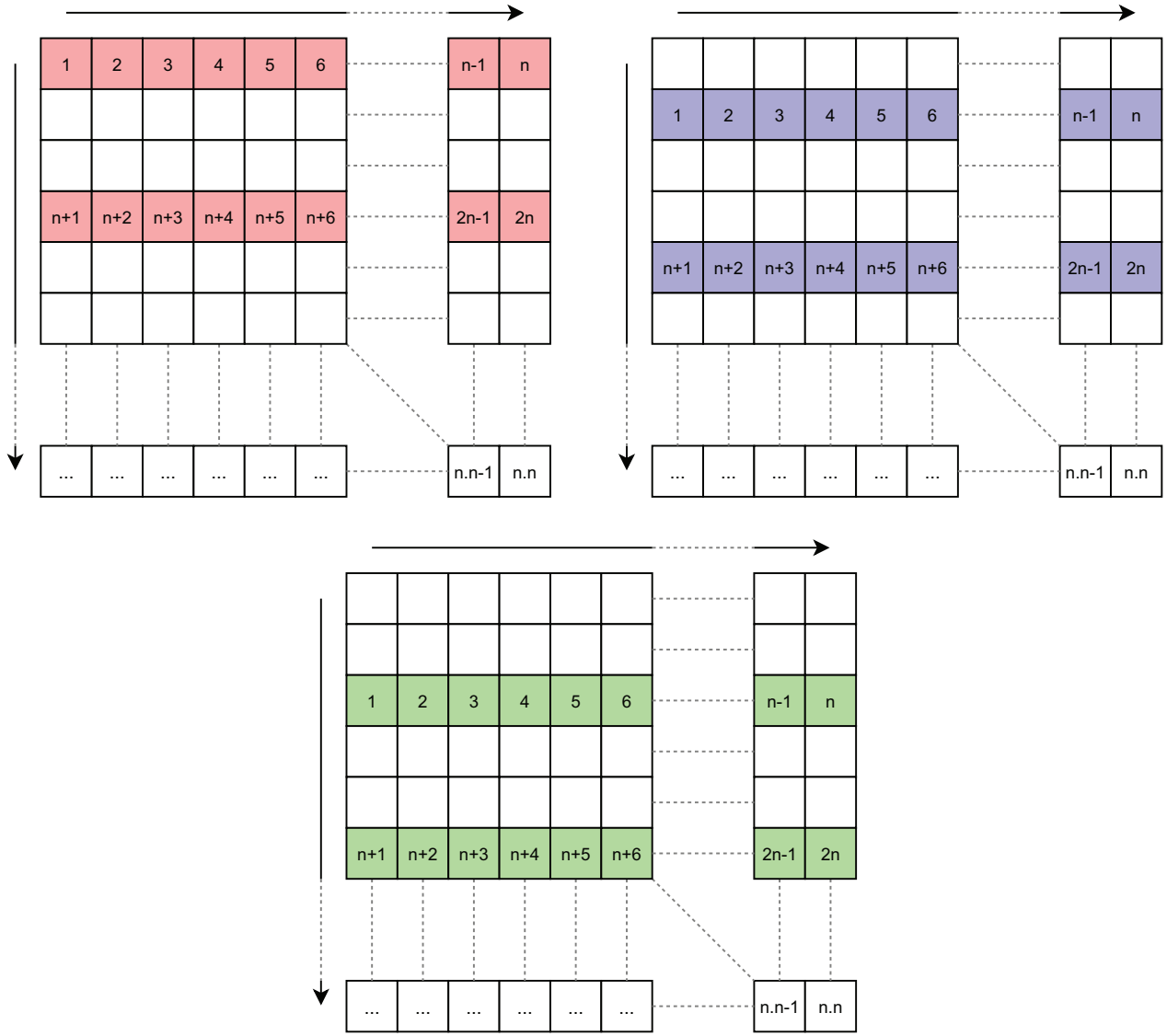
Zpracování po bloku rozptýlených řádků

Algoritmus řízení

```
for s=1 to STEPS:
    broadcastSend(CA1)
    for p=1 to numberOfProcs:
        CA3 = receive(myCA2)
        i = p-1
        while i < N:
            CA2[i] = CA3[i]
            i += shift
    CA1, CA2 = CA2, CA1
```

Algoritmus výpočtu

```
for s=1 to STEPS:
    myCA1 = broadcastReceive(CA1)
    i = rank-1
    while i < N:
        for j=0 to N-1:
            myCA2[i, j] = newVal(i, j)
        i += shift
    send(myCA2)
```

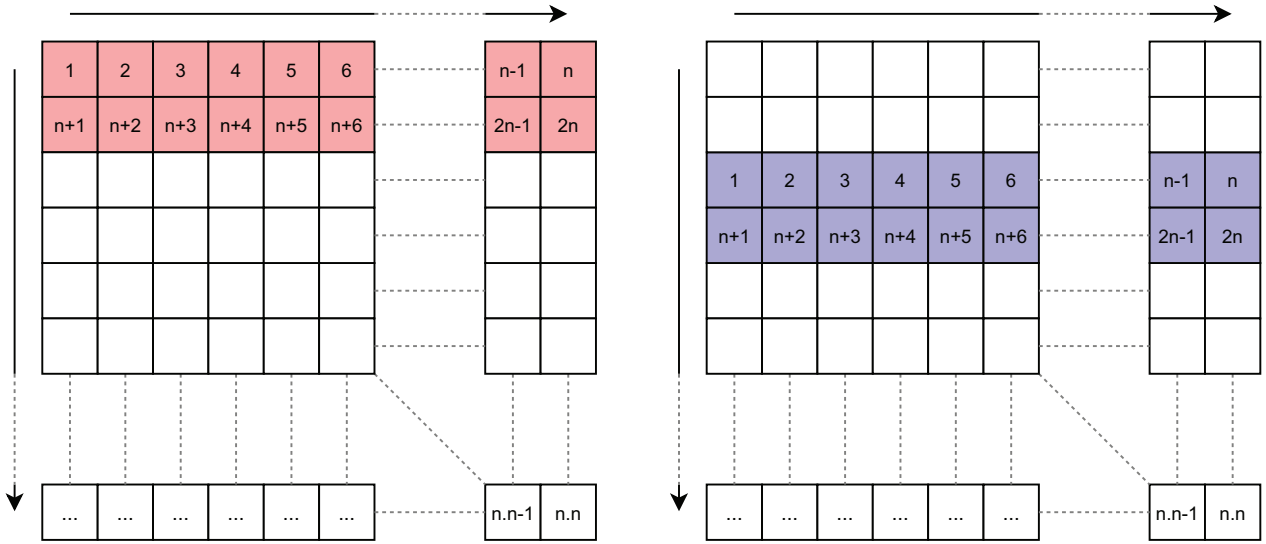


Zpracování po bloku rozptýlených řádků

Zpracování po souvislých blocích řádků

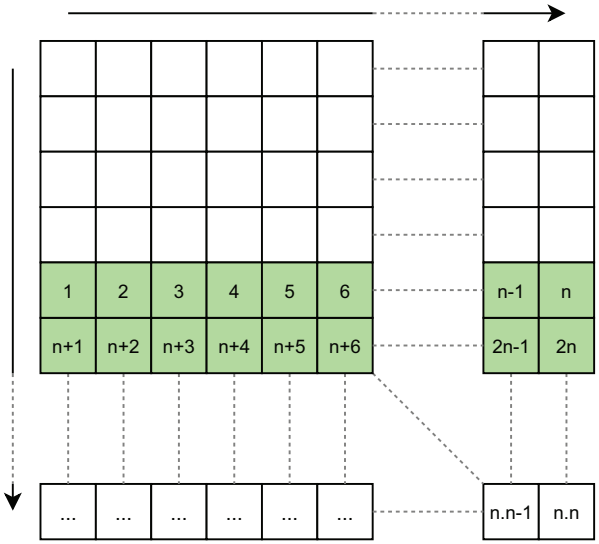
Algoritmus řízení

for s=1 **to** STEPS:
  broadcastSend(CA1)
  for p=1 **to** nOfProcs:
    iFr = (p-1)·(N/(nOfProcs-1))
    iTo = p·(N/(nOfProcs-1))
    CA2[iFr:iTo] = recv(myCA2[iFr:iTo])
  CA1, CA2 = CA2, CA1



Algoritmus výpočtu

for s=1 **to** STEPS:
  myCA1 = broadcastReceive(CA1)
  i = iFr
  while i < iTo:
    for j=0 **to** N-1:
      myCA2[i, j] = newVal(i, j)
    i += 1
  send(myCA2[iFr:iTo])



Zpracování po souvislých blocích řádků

Testované rozměry a stupně paralelizace

Conwayova hra Life

- 10×10, 20×20, . . . , 500×500 buněk (krok 10)

Langtonova a Bylova smyčka

- 35×35, 50×50, . . . , 500×500 buněk (krok 15)

Sekvenční zpracování

- Běh pouze na jednom jádře jednoho z procesorů

Paralelní zpracování

- 6, 8, 10, (12 a 14) paralelních procesů (MPI procesorů)
- K dispozici pouze 8 fyzických jader

Paralelní zpracování využívá vždy 1 procesor jako řídící a ostatní procesory jako výpočetní.

Způsob testování

Počet testů

- Každá kombinace parametrů spuštěna 10krát

  [typ CA]×[rozměr]×[metoda výpočtu]×[stupeň paralelizace]

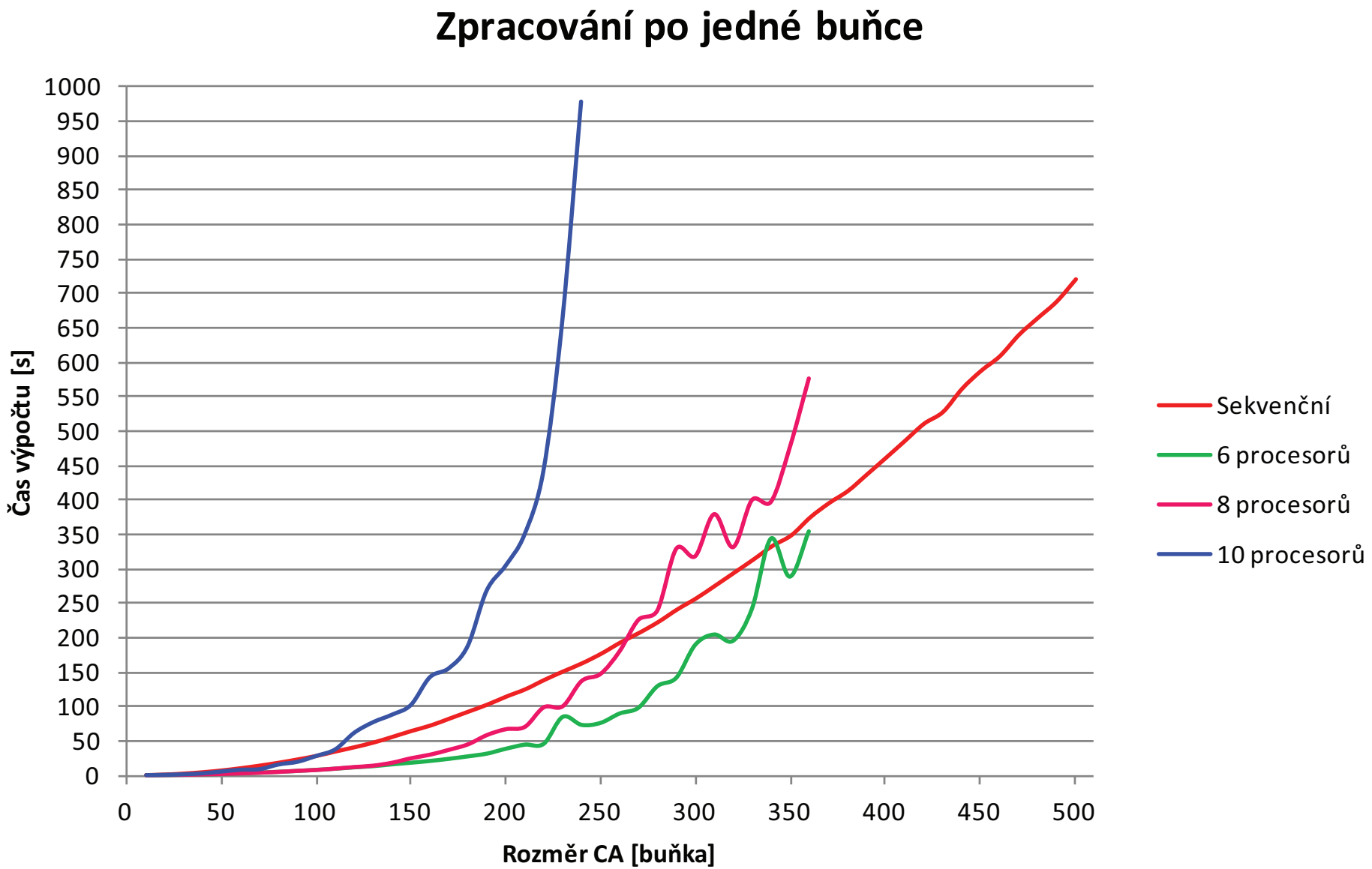
- V každém běhu proveden 100krát výpočet nových stavů celého CA

Měření délky trvání testu

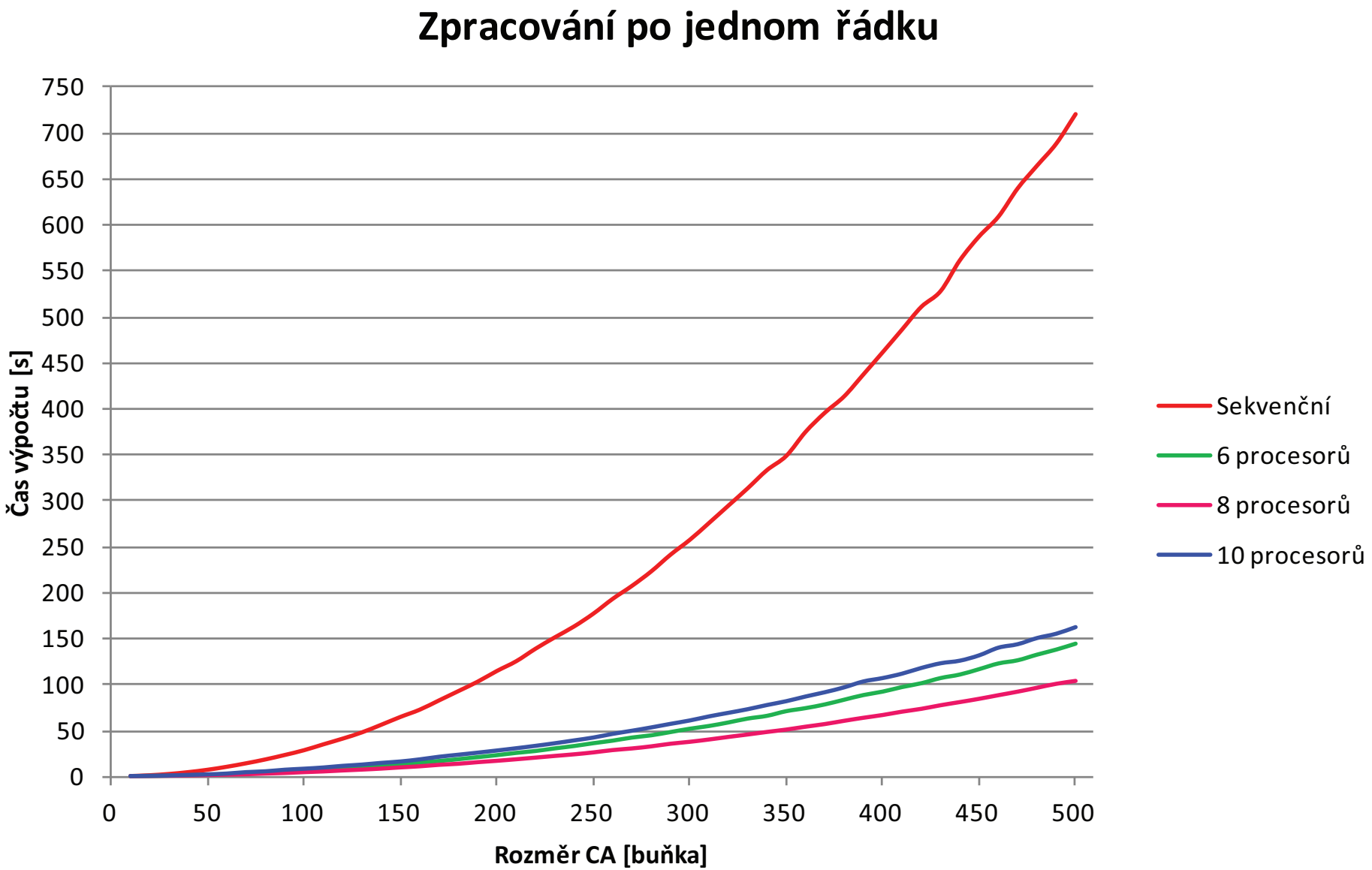
- Čas měřen pouze v řídícím procesu/procesoru
- Měření pomocí `time.process_time()` (sekvenční), `time.time()` (paralelní) a výpočet rozdílu dvou časů
- Délka běhu **nezahrnuje** inicializaci pole CA, pouze čas výpočtu

Výsledný čas jednoho testu je vypočten jako aritmetický průměr všech 10 časů změřených pro jednu kombinaci.

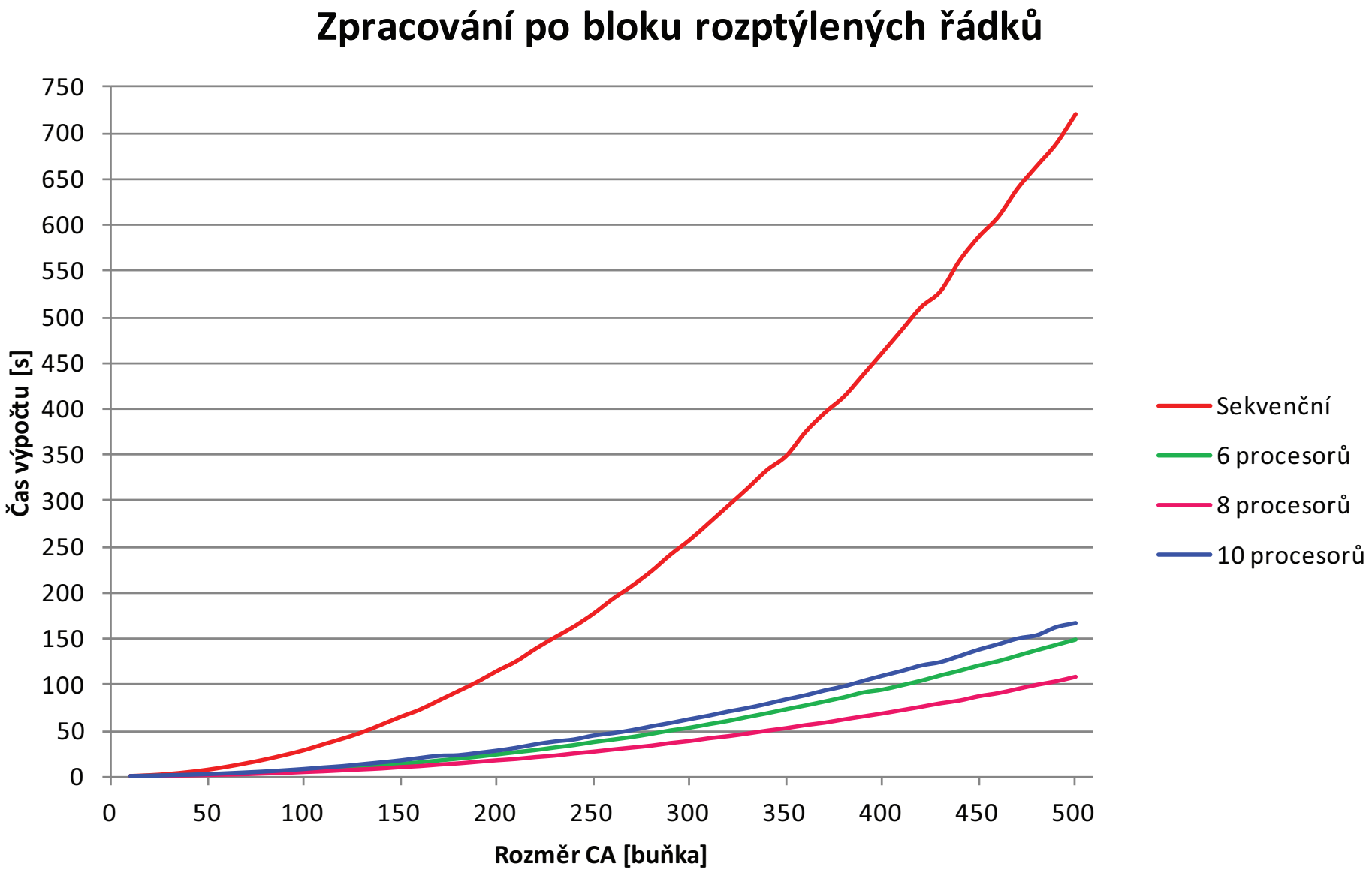
Výsledky – Hra Life



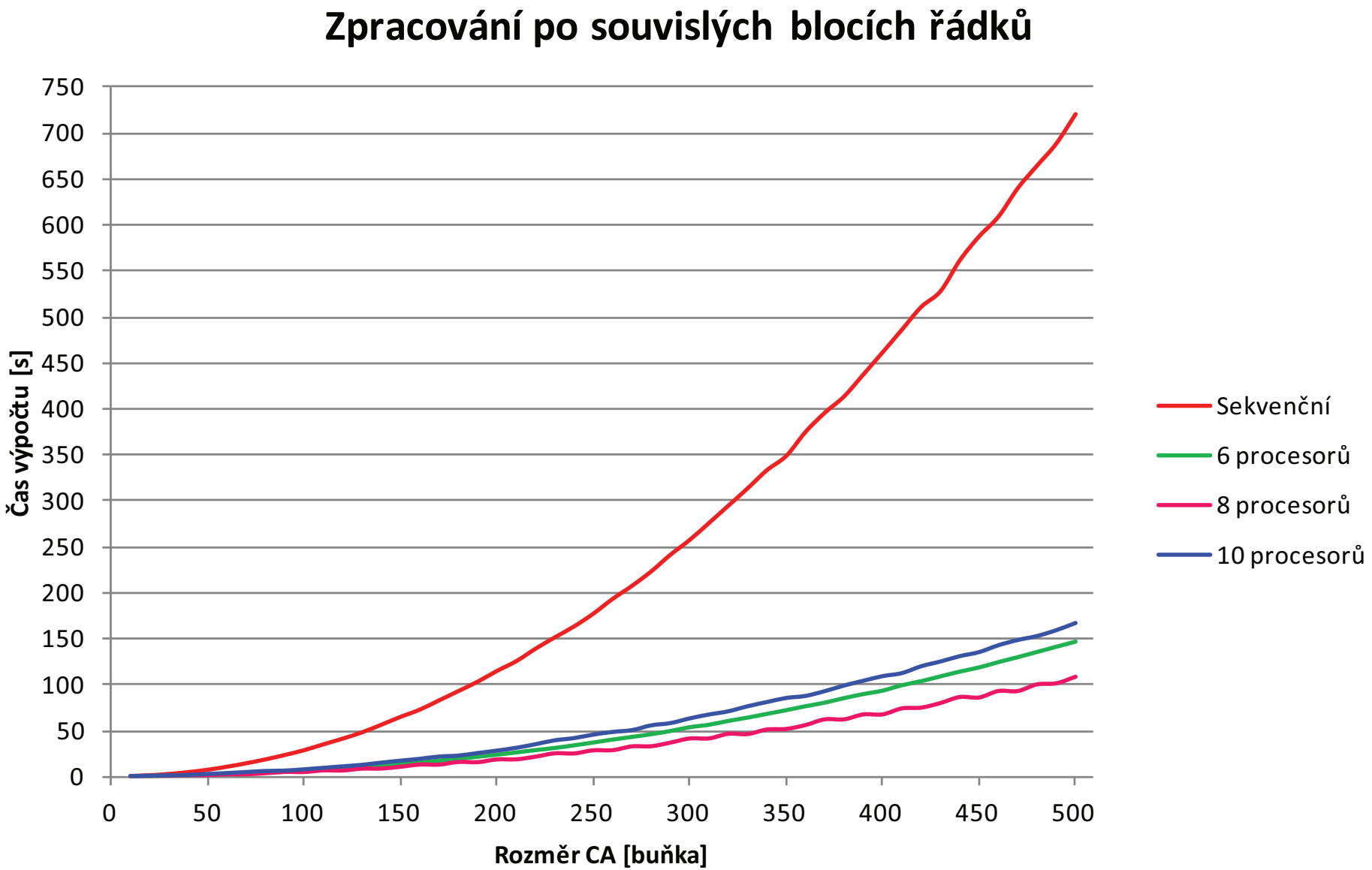
Výsledky – Hra Life



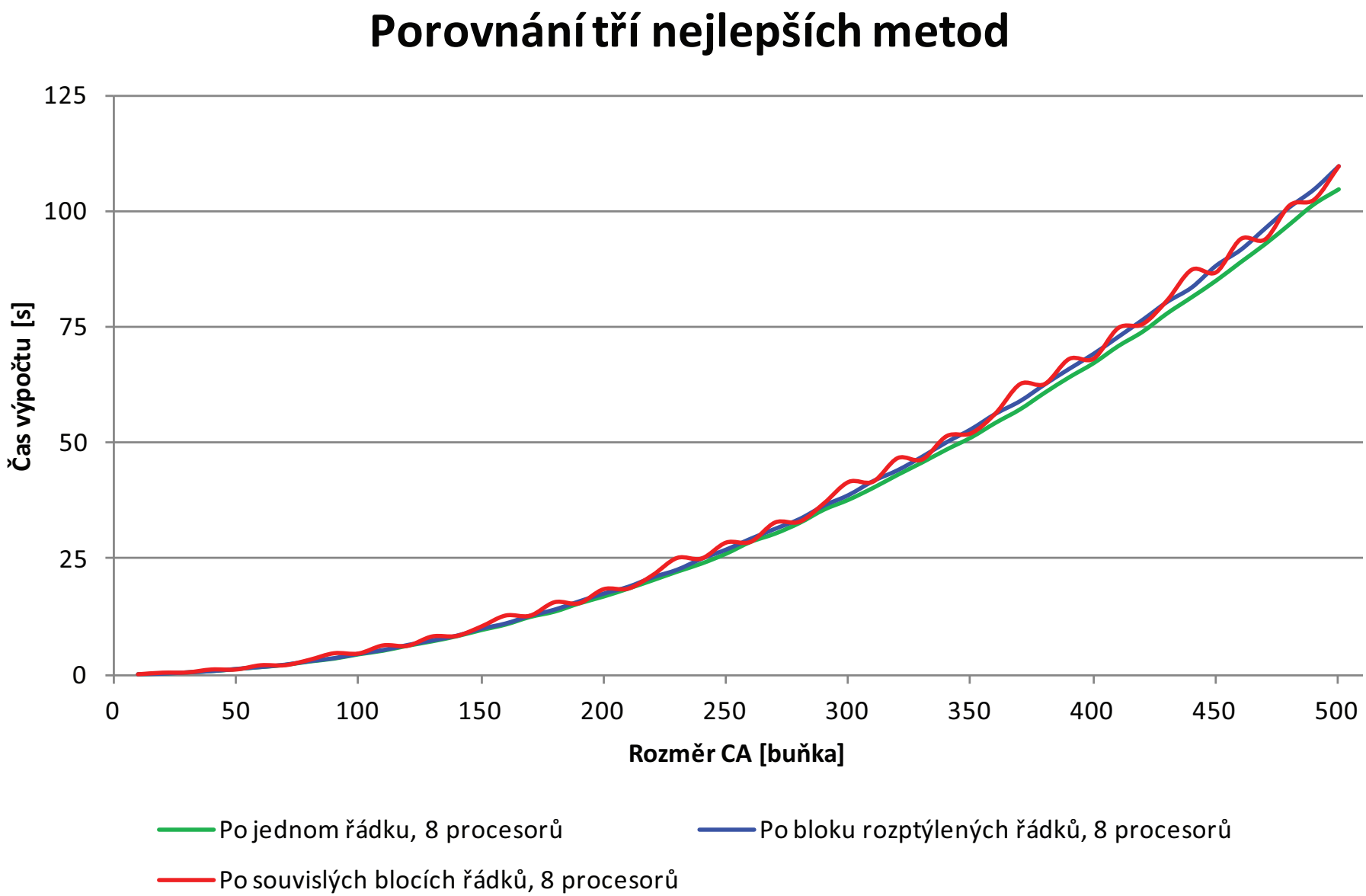
Výsledky – Hra Life



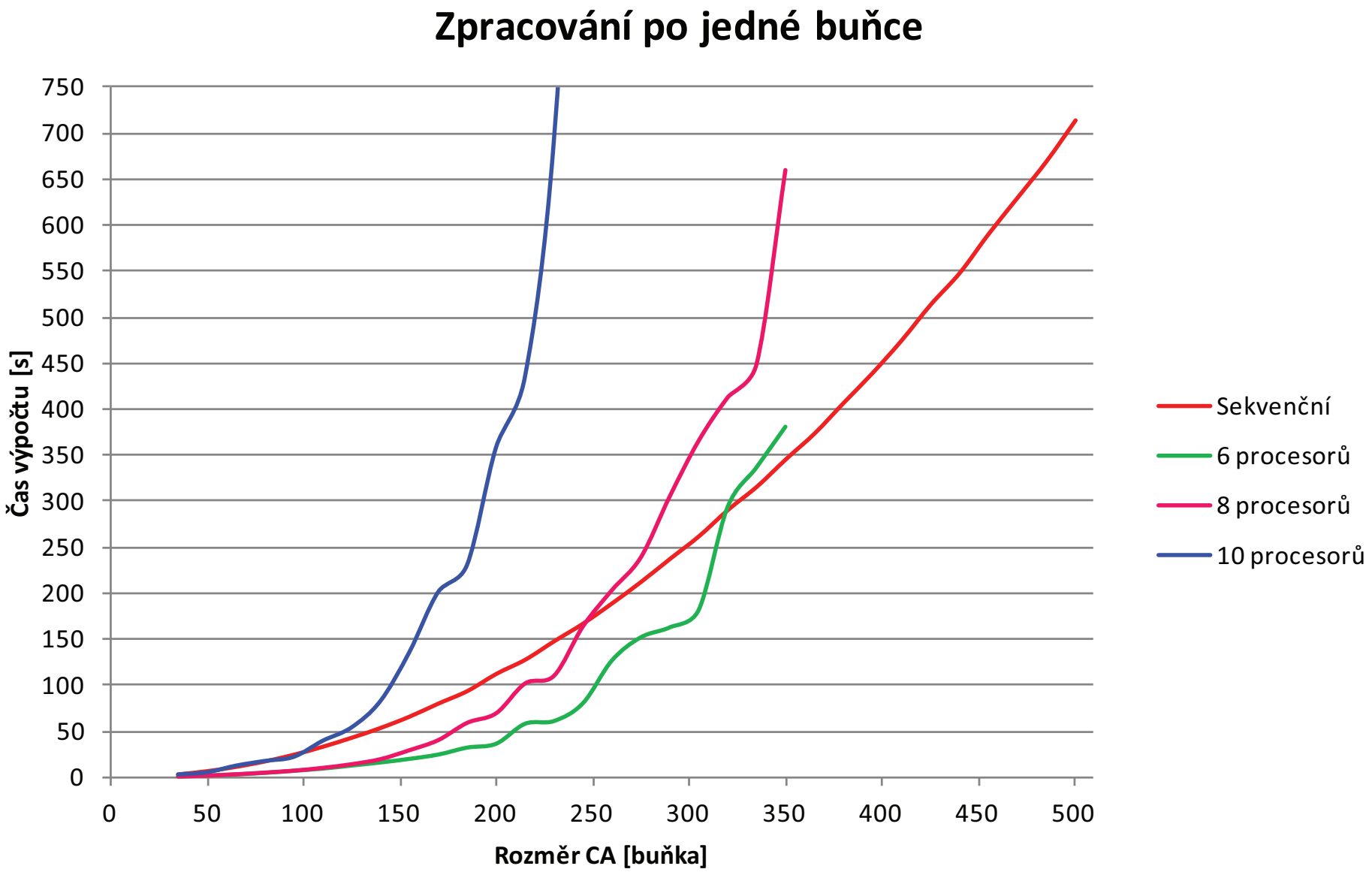
Výsledky – Hra Life



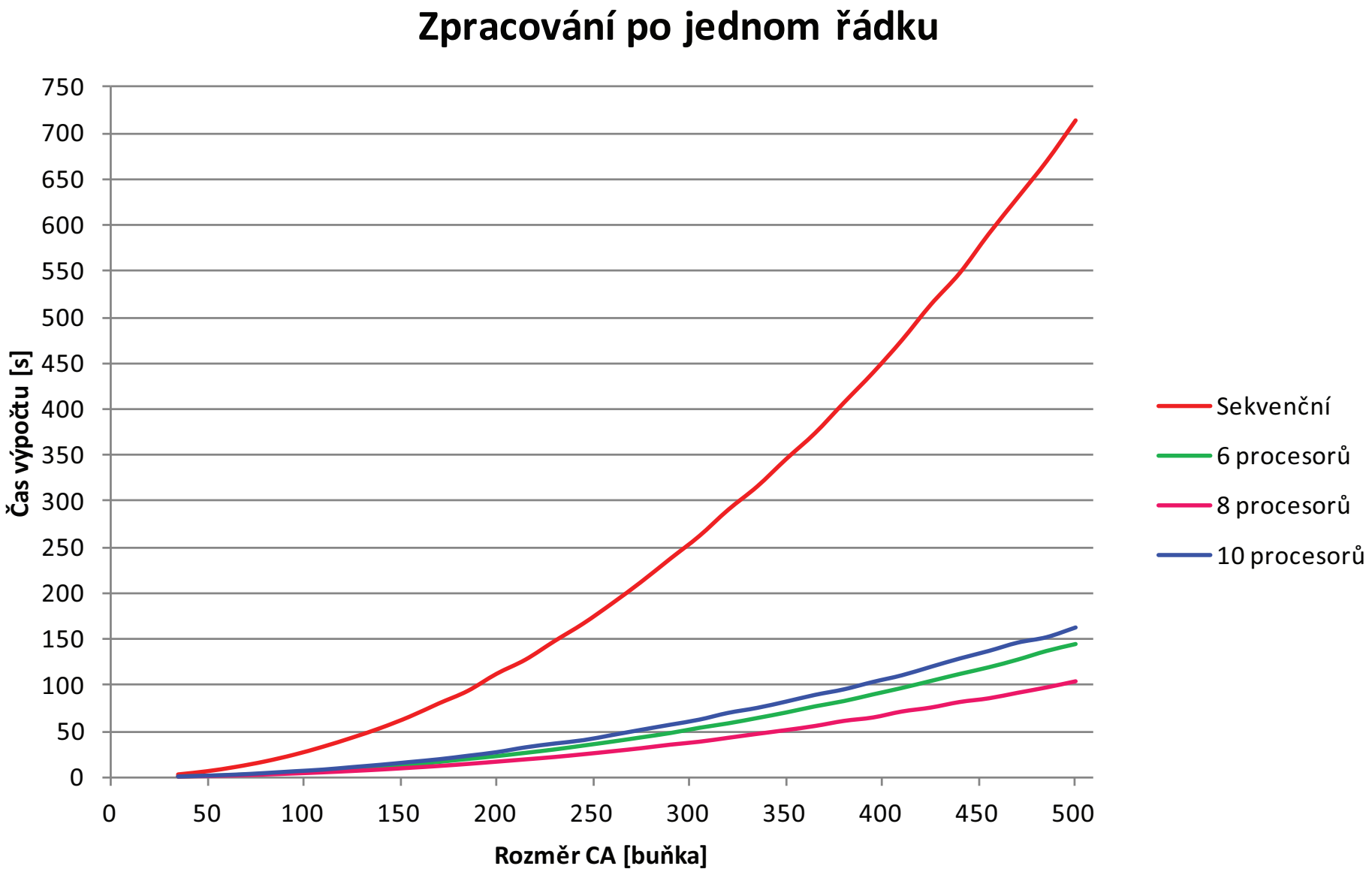
Výsledky – Hra Life



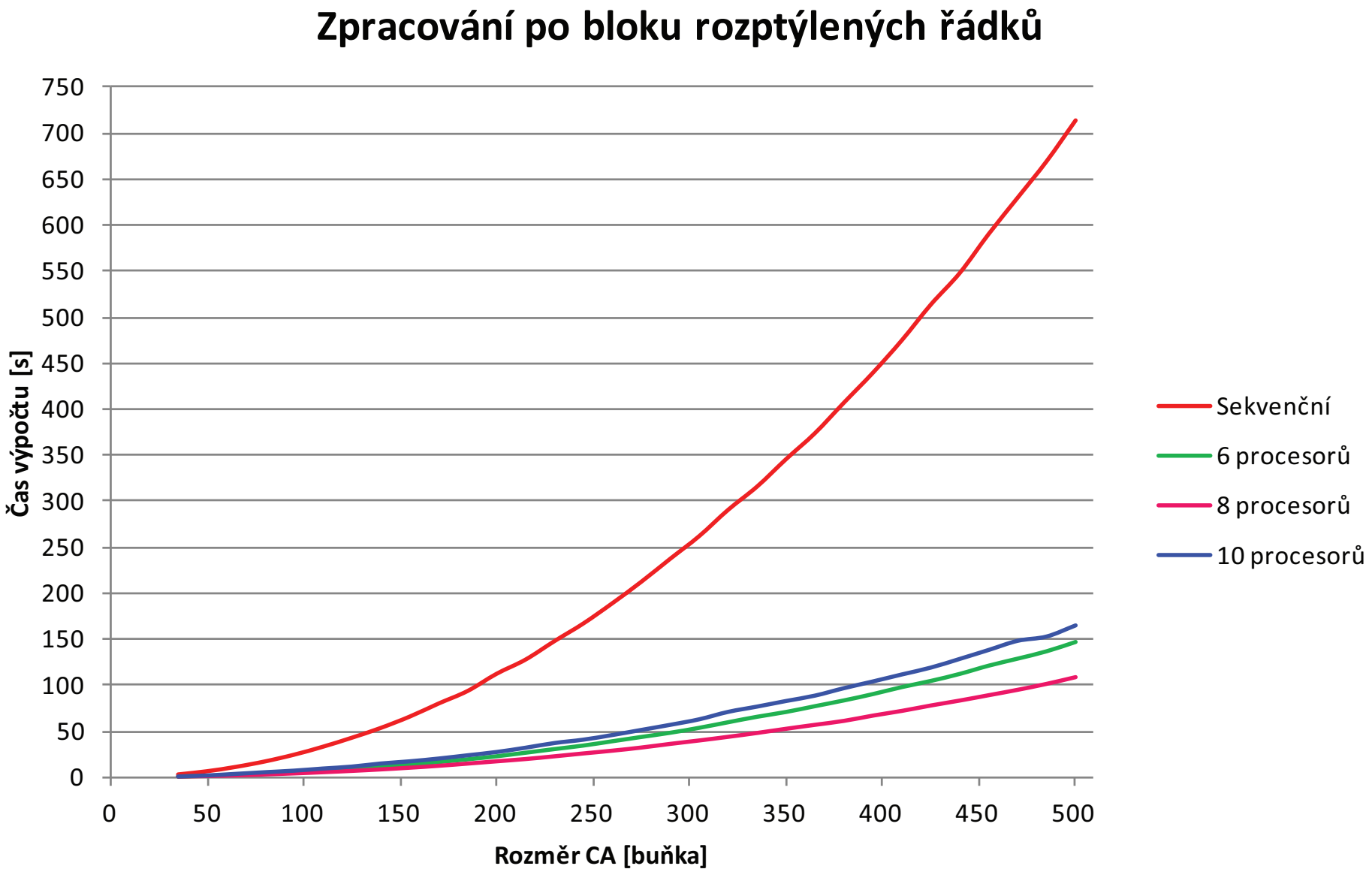
Výsledky – Langtonova smyčka



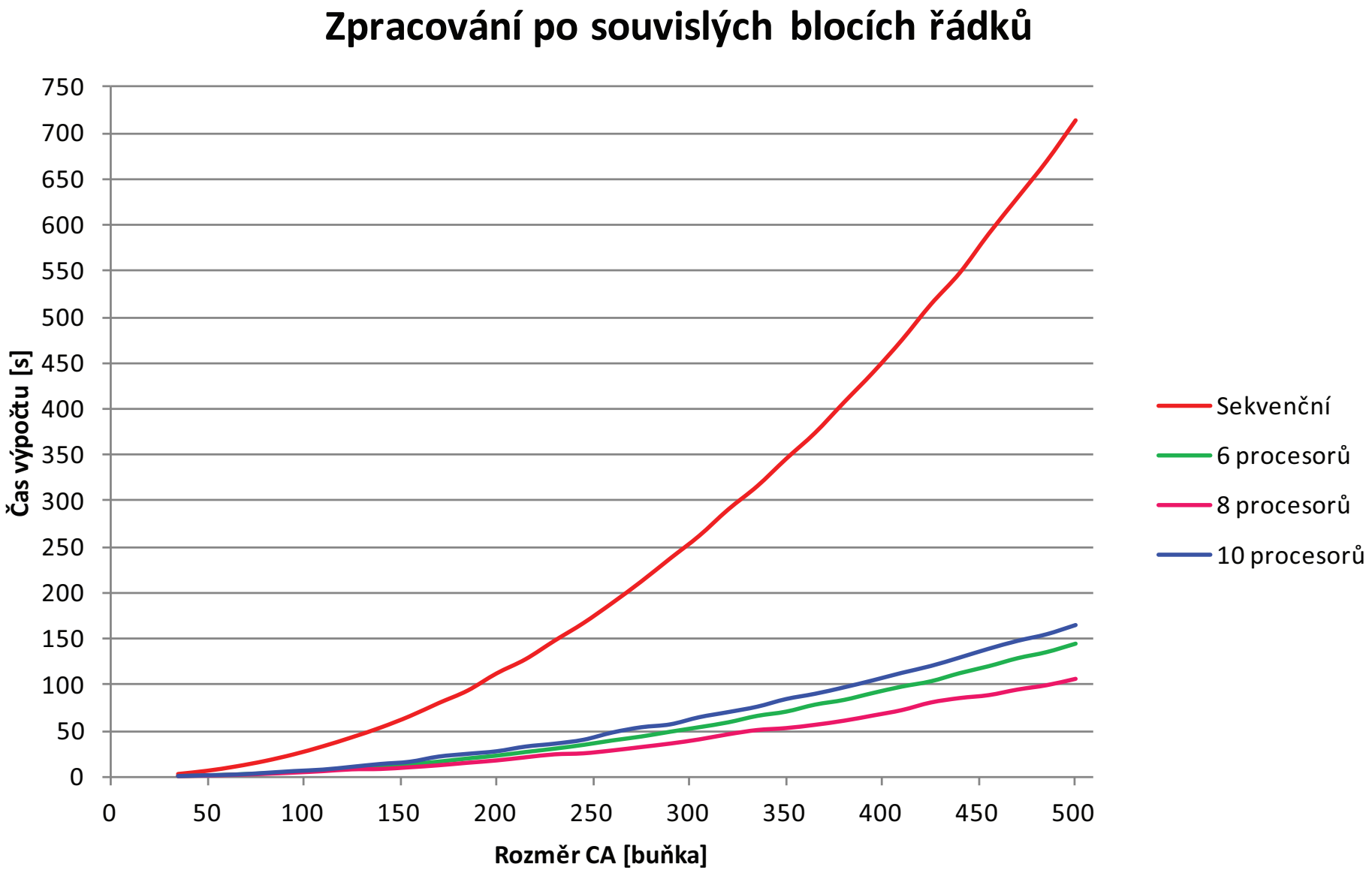
Výsledky – Langtonova smyčka



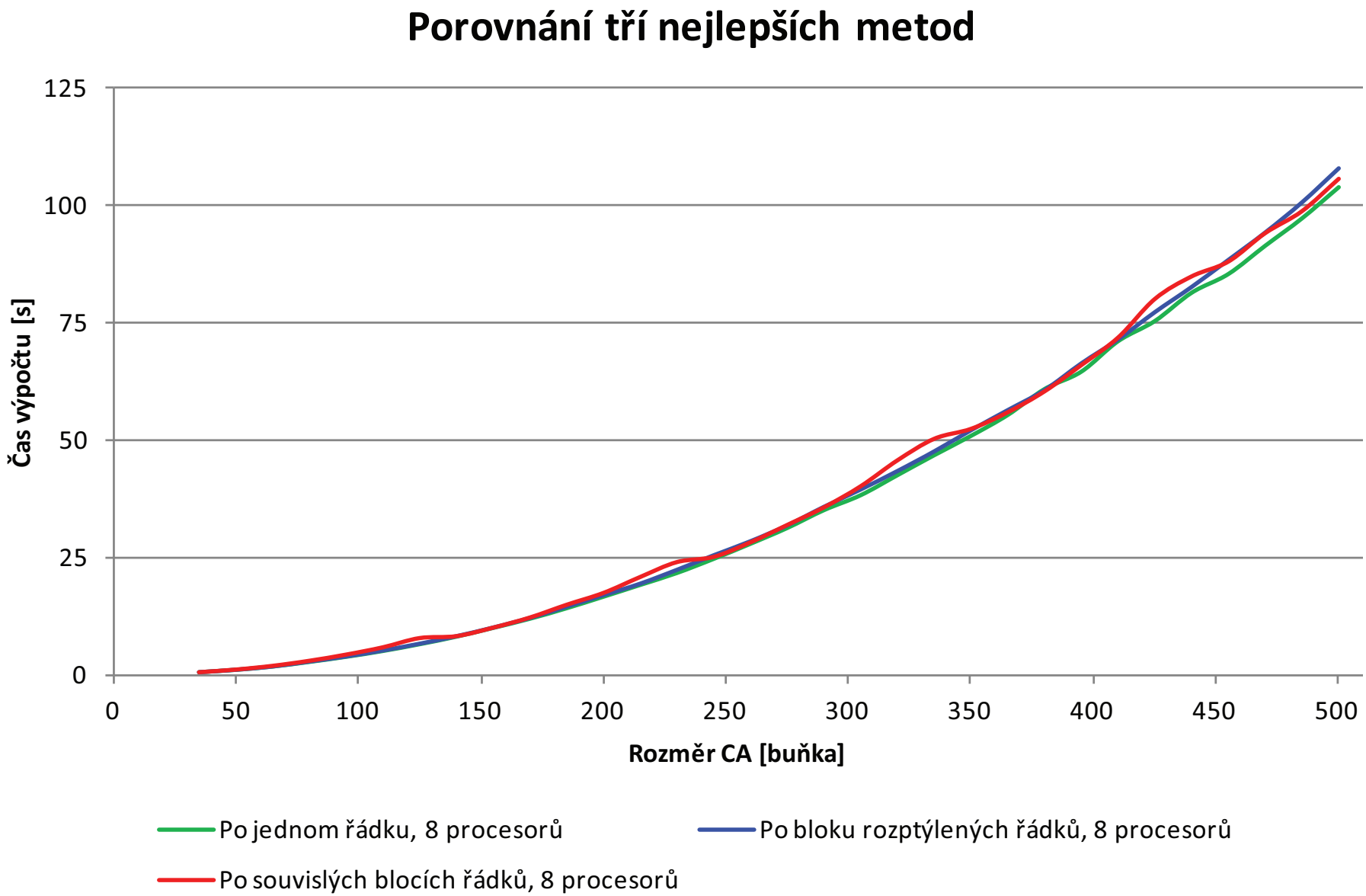
Výsledky – Langtonova smyčka



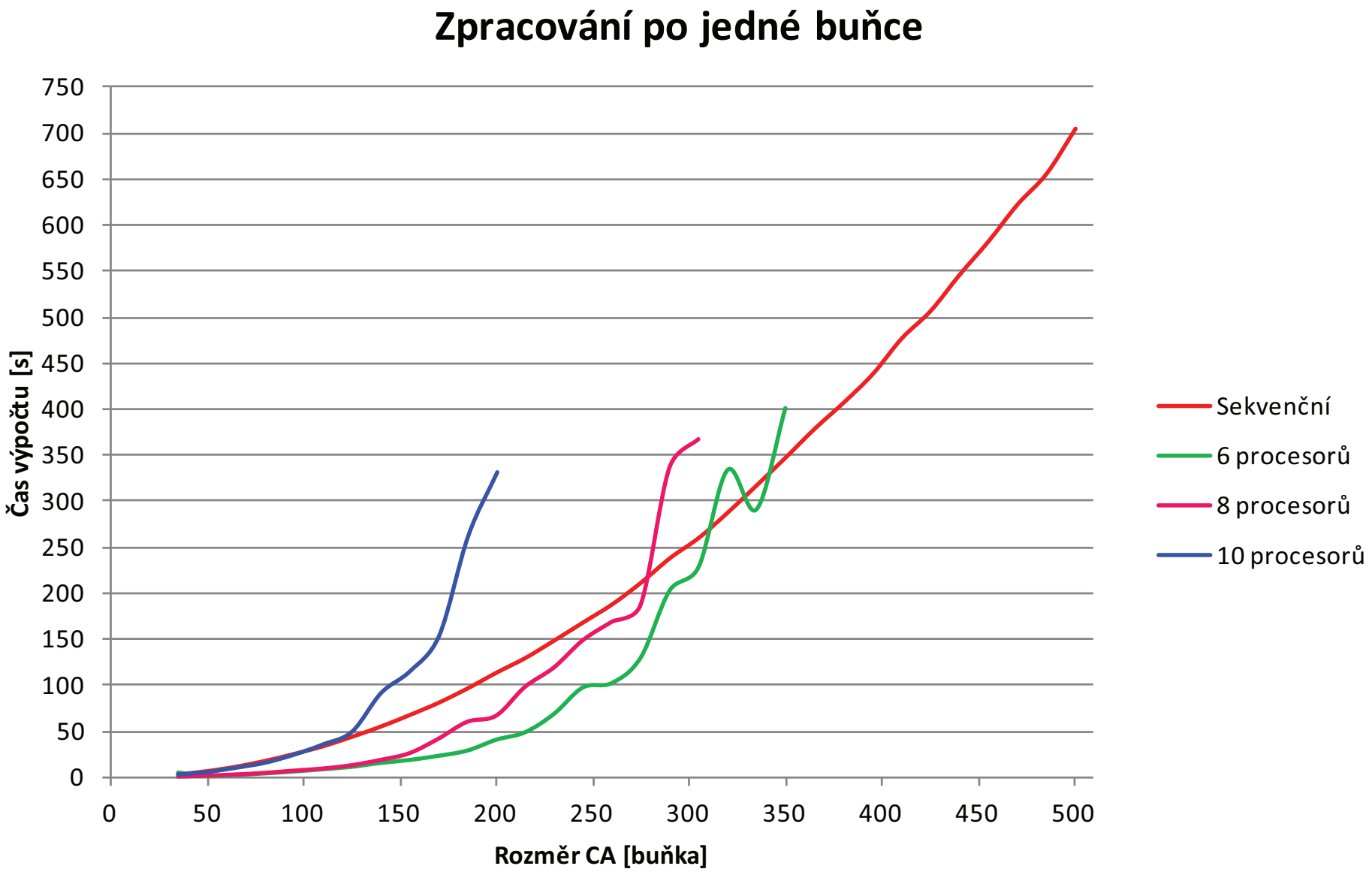
Výsledky – Langtonova smyčka



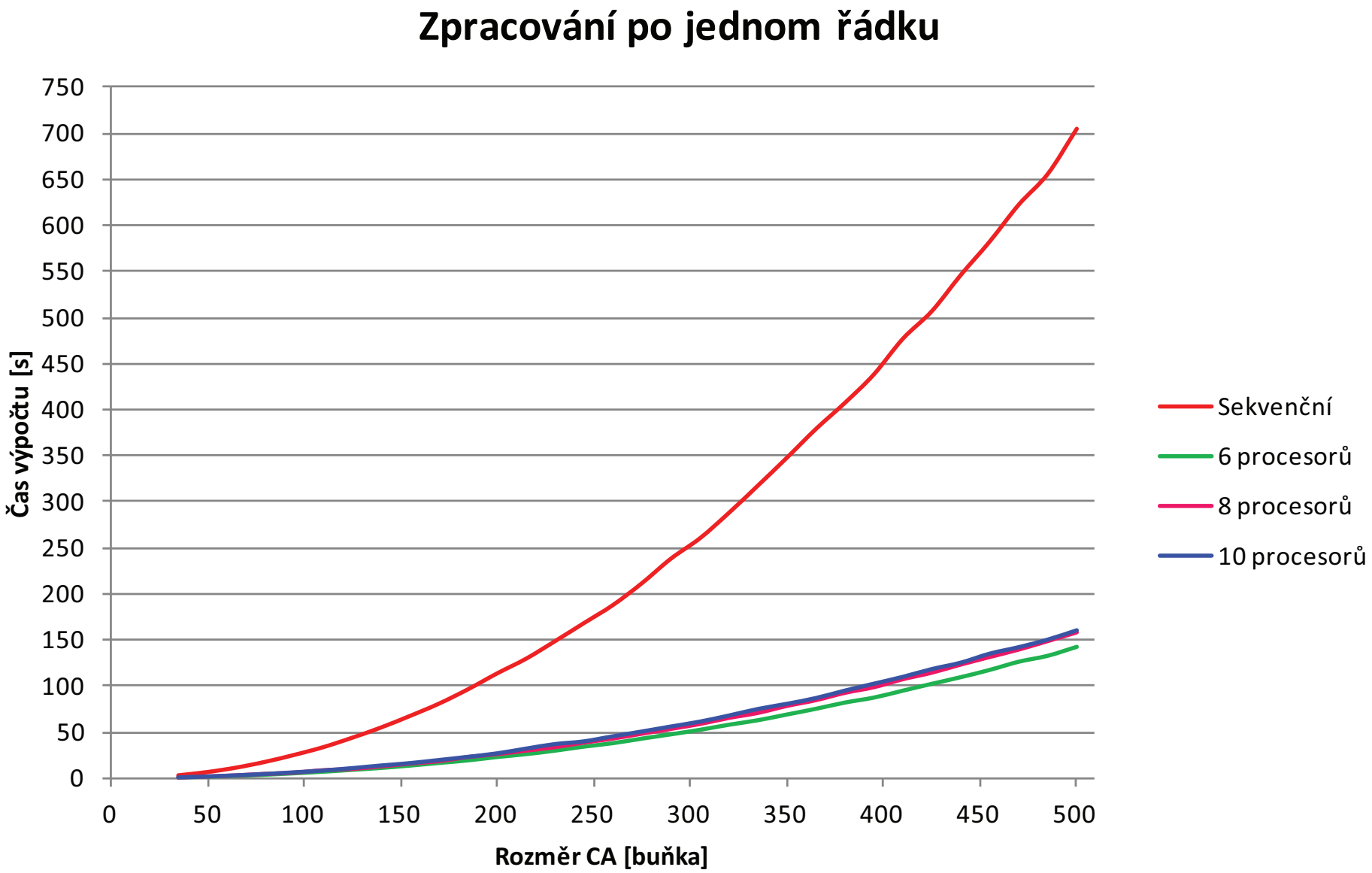
Výsledky – Langtonova smyčka



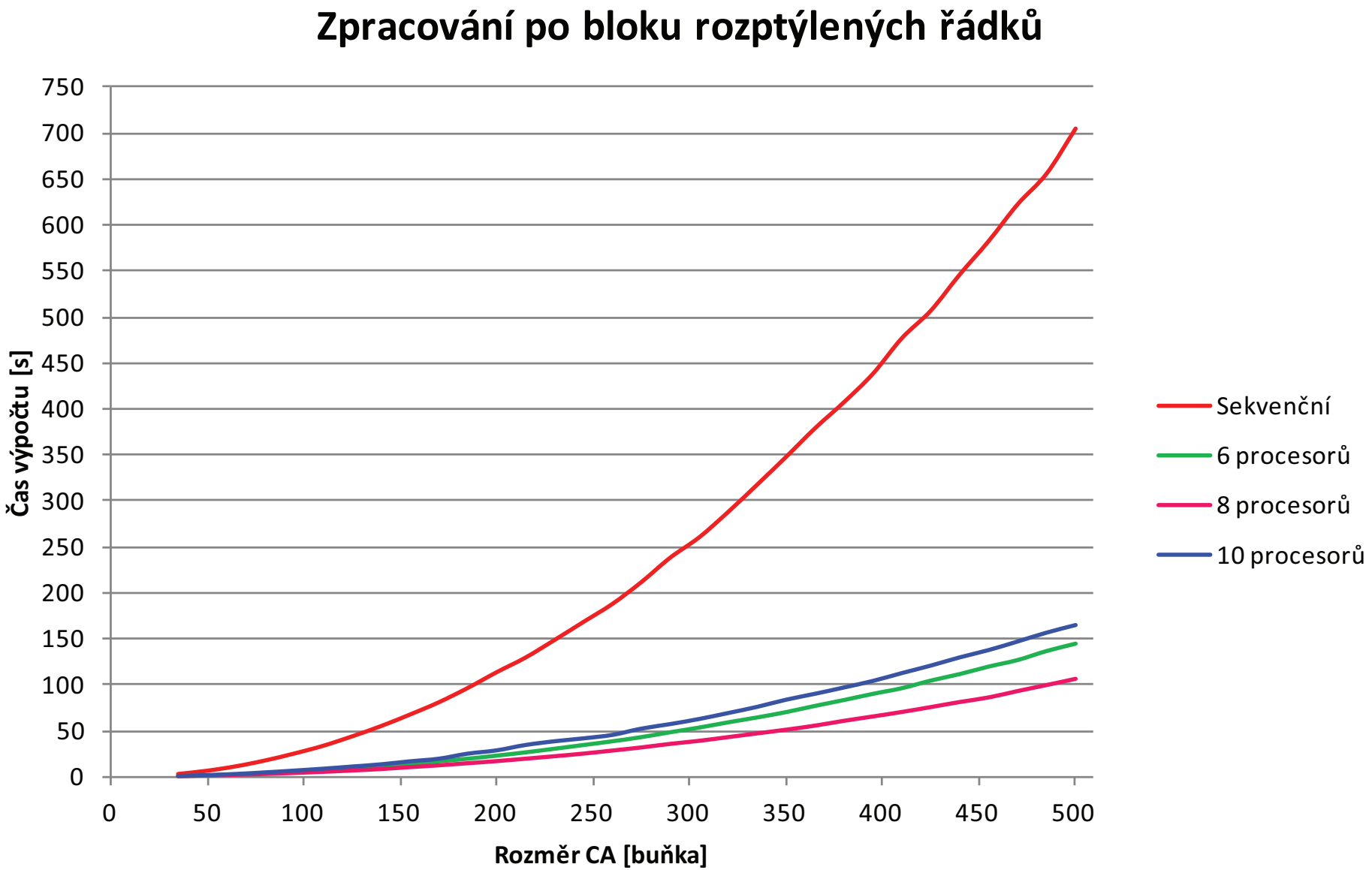
Výsledky – Bylova smyčka



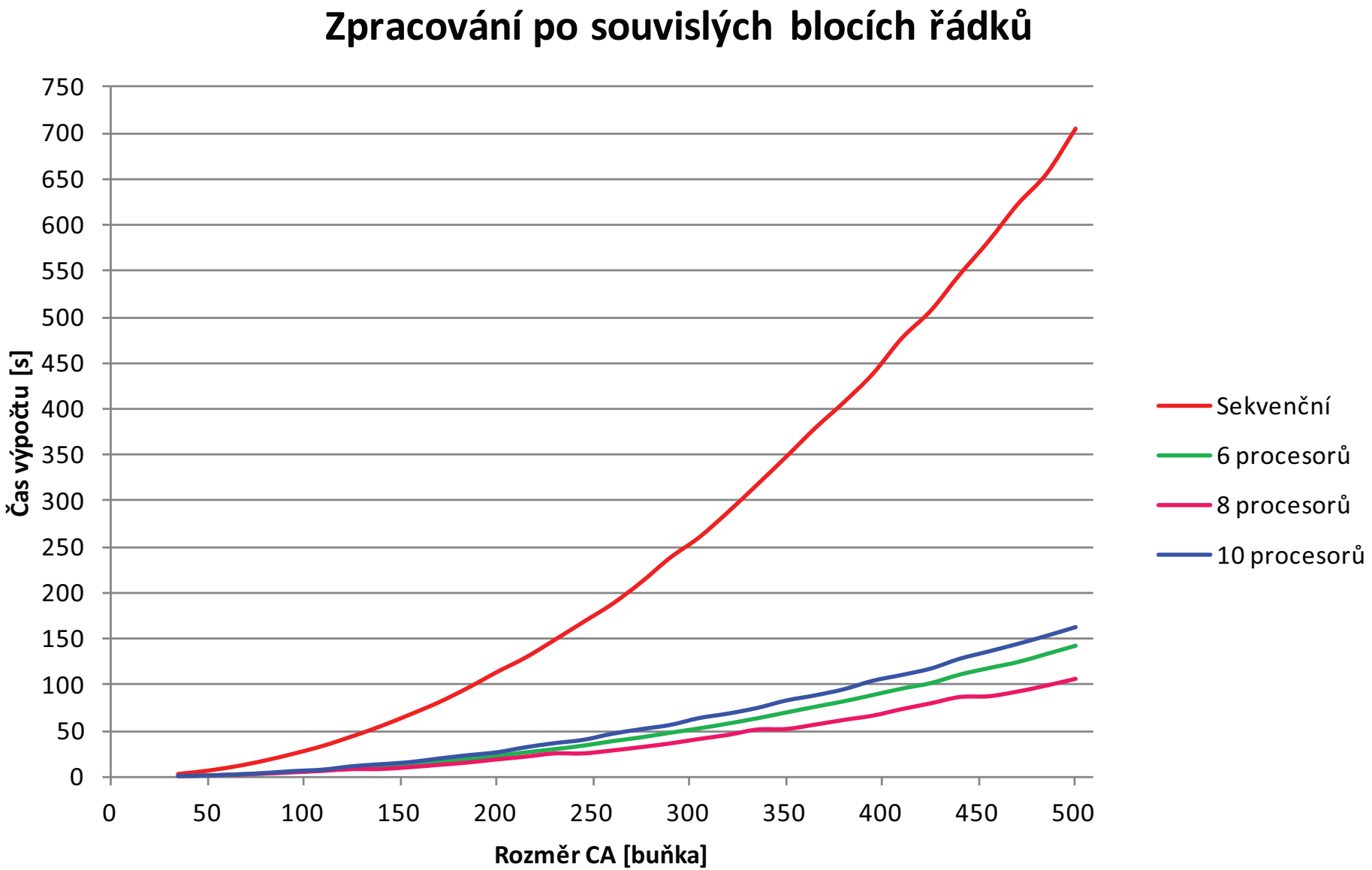
Výsledky – Bylova smyčka



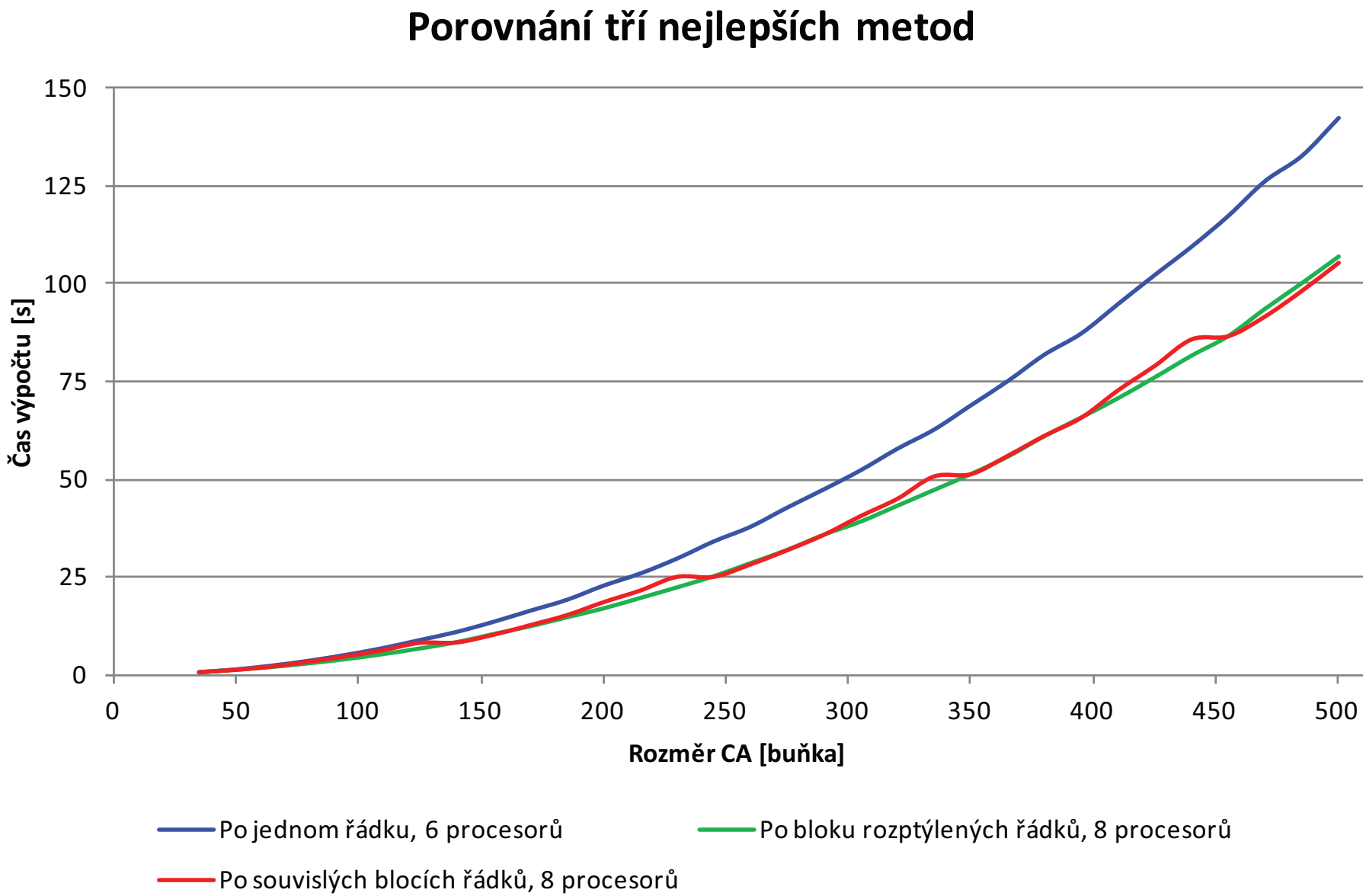
Výsledky – Bylova smyčka



Výsledky – Bylova smyčka



Výsledky – Bylova smyčka



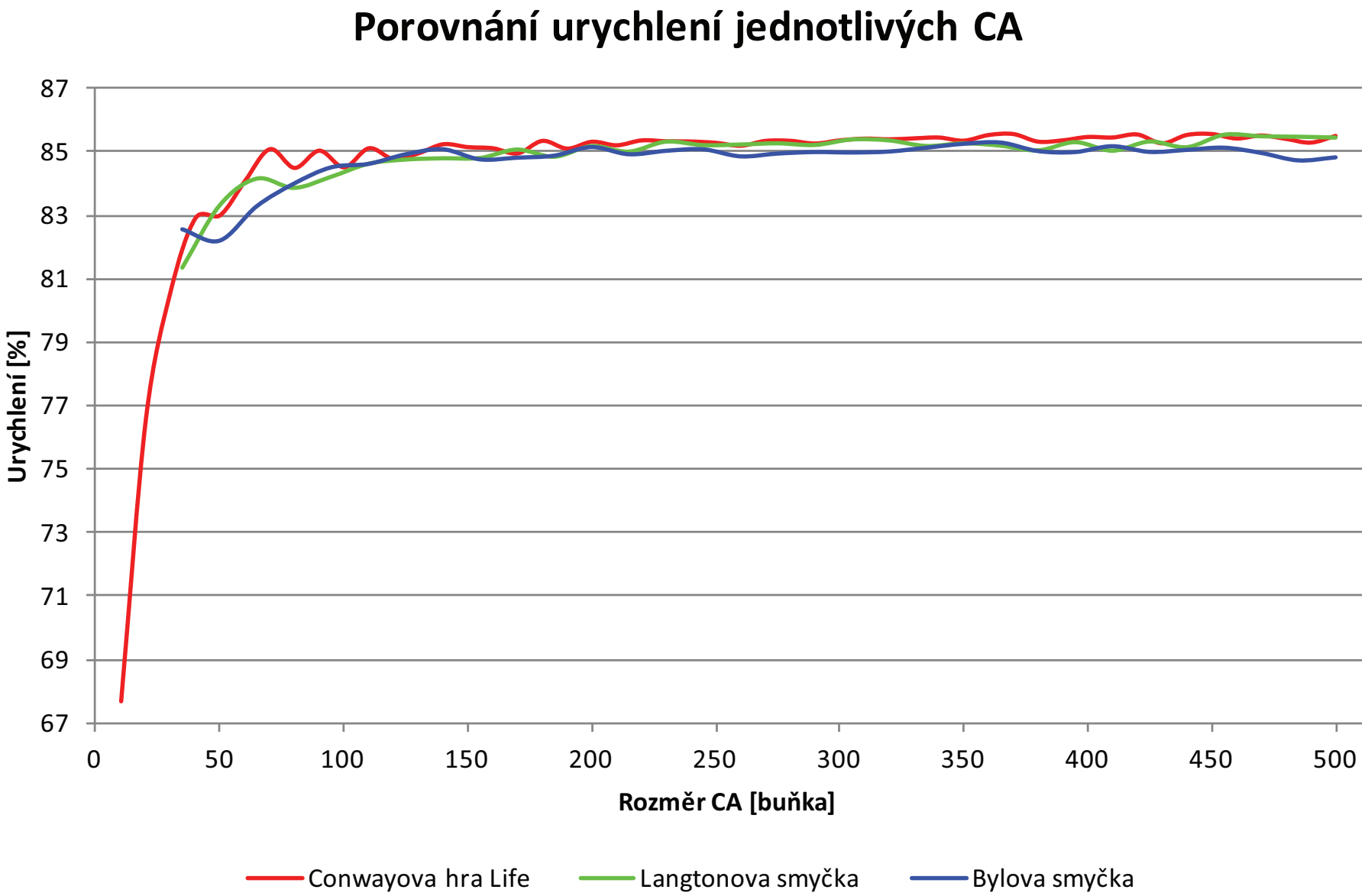
Výsledky – Urychlení výpočtu

Porovnání časů sekvenčního zpracování a paralelního výpočtu nejrychlejší metodou vybraných rozměrů CA

Rozměr CA	Conwayova hra Life		Langtonova smyčka		Bylova smyčka	
	Sekvenční zpracování	Nejrychlejší metoda	Sekvenční zpracování	Nejrychlejší metoda	Sekvenční zpracování	Nejrychlejší metoda
50	7,146	1,214	6,597	1,100	6,948	1,239
100/95	28,442	4,403	24,538	3,869	25,553	3,962
150/155	64,708	9,605	65,988	10,014	67,843	10,347
200	114,744	16,838	112,517	16,652	113,852	16,919
250/245	177,173	26,059	167,268	24,706	168,442	25,153
300/305	257,556	37,680	262,234	38,262	260,326	39,137
350	348,958	51,086	345,886	50,888	347,596	51,297
400/395	461,903	67,083	439,837	64,549	437,860	65,765
450/455	588,048	84,857	590,735	85,324	581,944	86,603
500	721,227	104,498	714,156	103,784	703,930	106,915

Tabulka časů výpočtů pro vybrané rozměry CA – v sekundách, hra Life/Langtonova a Bylova smyčka.

Výsledky – Urychlení výpočtu



Závěr

Paralelním výpočtem nových stavů CA pomocí knihovny *mpi4py* dosáhneme cca **85%** zrychlení oproti sekvenčnímu zpracování, což je vynikající výsledek.

Výhody:

+ rychlá implementace v Pythonu,
+ velké urychlení výpočtu.

Nevýhody:

- nárůst využitého paměťového prostoru,
- ?

Knihovna *mpi4py* je použitelná pro implementaci paralelních výpočtů nových stavů CA a lze ji pro tento účel rozhodně doporučit.