პარალელური პროგრამირების ტექნოლოგიები

ჯგუფური პროექტი

სოფიო ჩიქვინიძე
მარიამ დოლიაშვილი

ხელმძღვანელი:

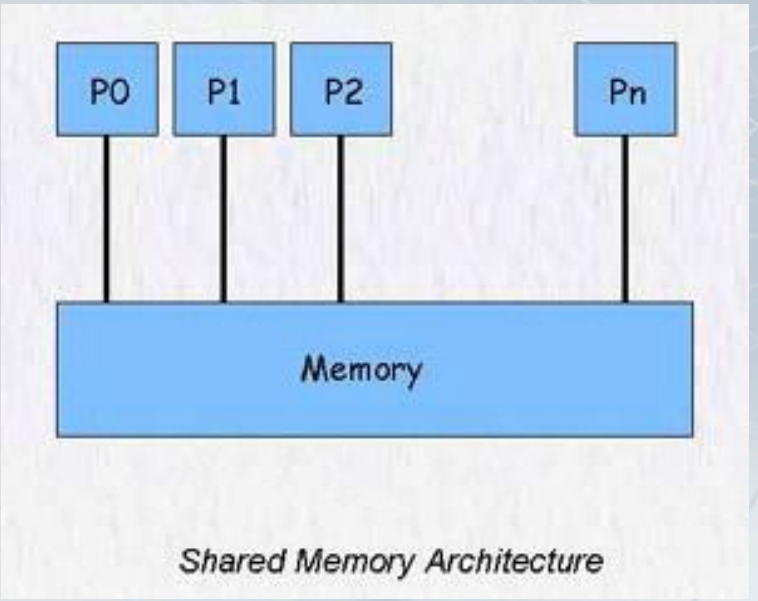
სრული პროფესორი კობა გელაშვილი

შინაარსი

- მეხსიერების საზიარო მოდელი
  - მატრიცების გადამრავლება
  - Merge sort
  - გამოყენებული ტექნოლოგიები
  - შედეგები
- მეხსიერების განაწილებული მოდელი
  - Message Passing Interface (MPI)
  - მატრიცების გადამრავლება
  - Merge sort
  - შედეგები
- Thread-safe stack

მეხსიერების საზიარო მოდელი

- Multithreading-ის შესაძლებლობა
- ძირითადი ბრძანებები:
  - Spawn
  - Sync
  - parallel



მატრიცების გადამრავლება

```
P-MERGE(T, p1, r1, p2, r2, A, p3)
 1  n1 = r1 − p1 + 1
 2  n2 = r2 − p2 + 1
 3  if n1 < n2
 4       exchange p1 with p2
 5       exchange r1 with r2
 6       exchange n1 with n2
 7  if n1 == 0
 8       return
 9  else q1 = ⌊(p1 + r1)/2⌋
10       q2 = BINARY-SEARCH(T[q1], T, p2, r2)
11       q3 = p3 + (q1 − p1) + (q2 − p2)
12       A[q3] = T[q1]
13       spawn P-MERGE(T, p1, q1 − 1, p2, q2 − 1, A, p3)
14       P-MERGE(T, q1 + 1, r1, q2, r2, A, q3 + 1)
15       sync
```

Merge Sort

```
P-MERGE-SORT(A, p, r, B, s)
 1  n = r − p + 1
 2  if n == 1
 3       B[s] = A[p]
 4  else let T[1..n] be a new array
 5       q = ⌊(p + r)/2⌋
 6       q' = q − p + 1
 7       spawn P-MERGE-SORT(A, p, q, T, 1)
 8       P-MERGE-SORT(A, q + 1, r, T, q' + 1)
 9       sync
10       P-MERGE(T, 1, q', q' + 1, n, B, s)
```

გამოყენებული ტექნოლოგიები

Microsoft PPL (Parallel Patterns Library)

- ჩაშენებულია Visual Studio-ში
- Parallel_for
- Thread-ების რაოდენობის კონტროლი

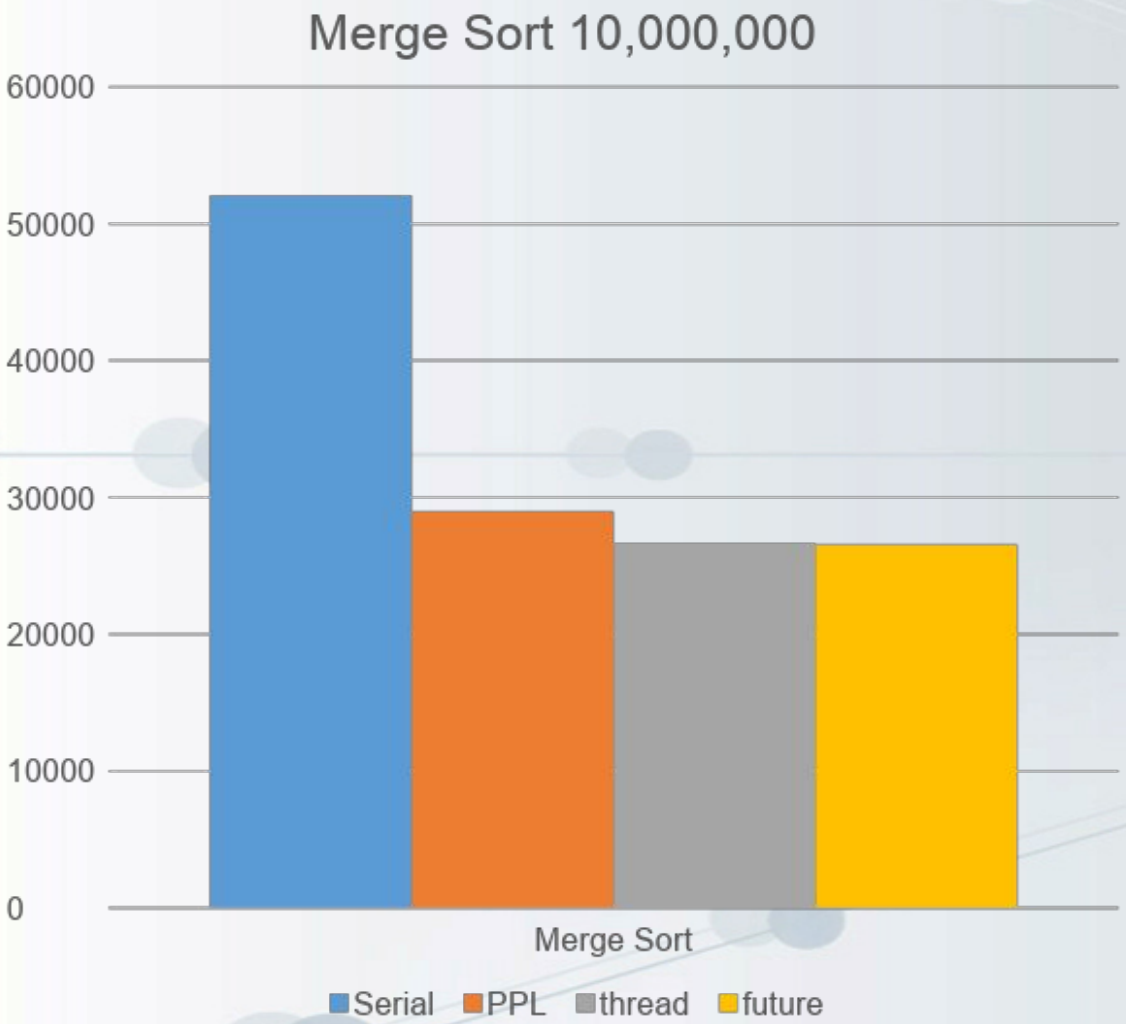
C++ 11 Threads

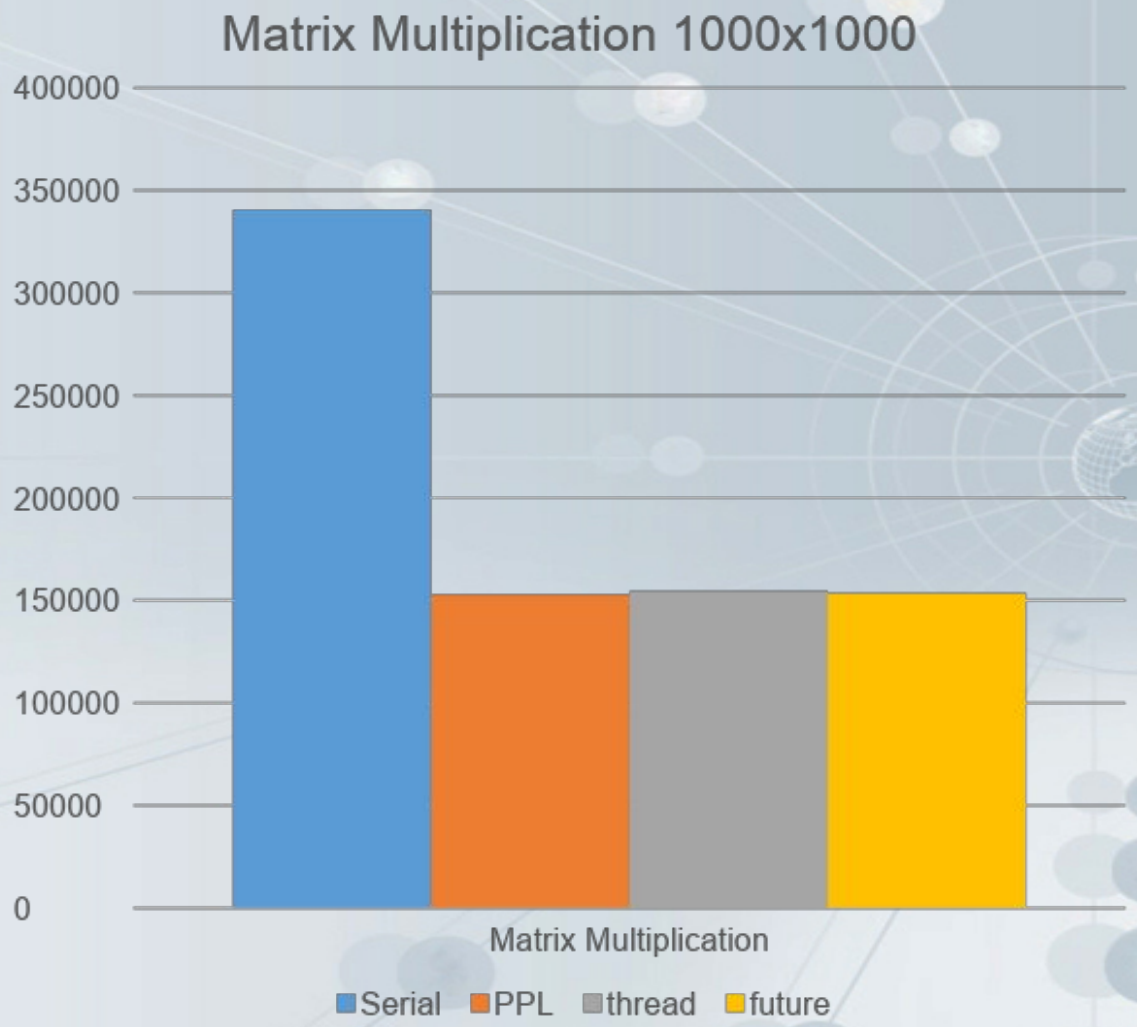
- Cross-platform
- დაბალი დონის აბსტრაქცია
- Std::hardware_concurrency()

C++ 11 Future

- Cross-platform
- მაღალი დონის აბსტრაქცია
- მეტი ფუნქციონალი

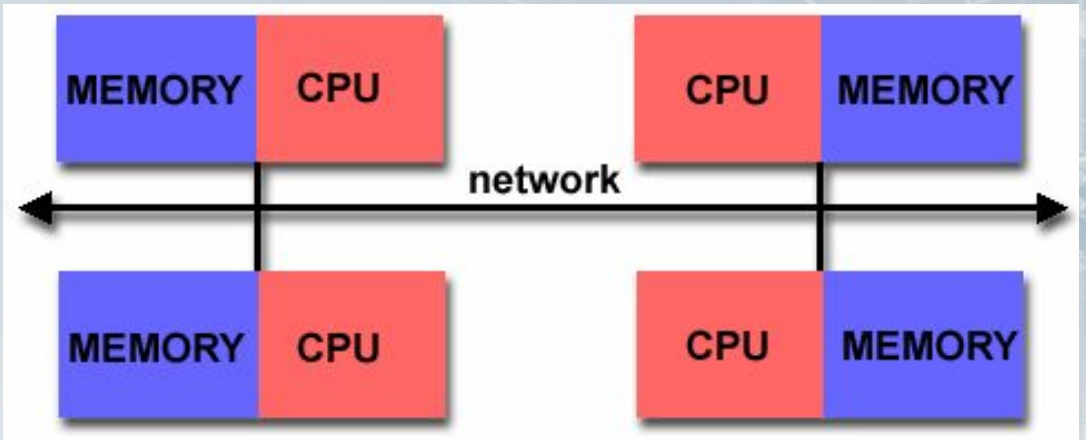
მიღებული შედეგების ანალიზი (Intel Core i7-3537U)





მეხსიერების განაწილებული მოდელი

- პროცესები thread-ების ნაცვლად
- პროცესებს შორის კომუნიკაცია მესიჯების მეშვეობით



Message Passing Interface (MPI)

- სტანდარტული საკომუნიკაციო პროტოკოლი
- MPI სტანდარტში არსებული ძირითადი ცნებები:
  - კომუნიკატორი
    - MPI_COMM_WORLD
    - MPI_COMM_SPLIT
  - Point-to-point კომუნიკაცია
    - MPI_Send
    - MPI_Recv
  - კოლექტიური კომუნიკაცია
    - MPI_Bcast
    - MPI_Reduce

მატრიცების გადამრავლება

- Root პროცესი:
  - მონაცემების ინიციალიზაცია
  - პირველი მატრიცის დაყოფა და გადანაწილება სხვა პროცესებზე
  - მეორე მატრიცის Broadcast
- დამხმარე პროცესები:
  - მიღებული ინფორმაციის დამუშავება
  - შედეგების გაგზავნა root პროცესში

Merge Sort

- Root პროცესი:
  - მონაცემების ინიციალიზაცია
  - Merge sort-ის გამოძახება
- დამხმარე პროცესები:
  - მონაცემების მიღება
  - Merge sort-ის გამოძახება
  - დასორტირებული მონაცემების root-ში გაგზავნა
- Merge sort ფუნქცია:
  - მასივის პირველი ნაწილის გაგზავნა დამხმარე პროცესში
  - მეორე ნაწილის დასორტირება მიმდინარე პროცესში
  - Merge

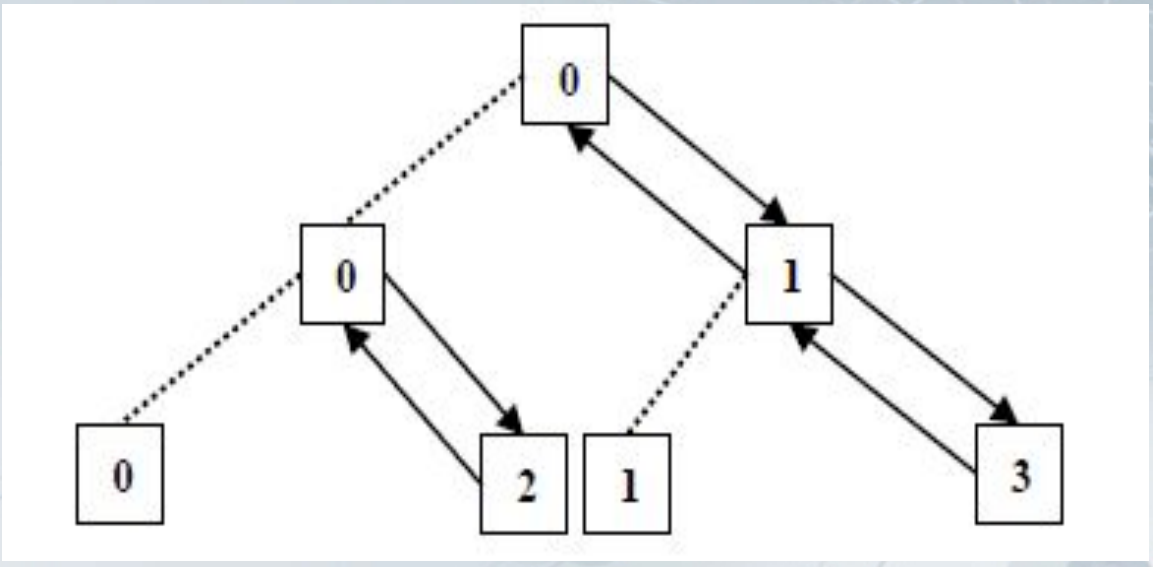
Merge sort-ის პროცესების ხე

- დამხმარე პროცესის იდენტიფიკატორის გამოთვლა:

```
int helper_id = my_id + pow(2, level);
```

- მიმდინარე level-ის გამოთვლა:

```
int my_topmost_level(int my_id) {
int level = 0;
while (pow(2, level) <= my_id) level++;
 return level;
}
```



შედეგები

	100 x 100	500 x 500	1000 x 1000
Serial	0.004767 s	0.757161 s	10.838827 s
Parallel	0.002998 s	0.668723 s	4.967020 s

მატრიცების გადამრავლება

	1000	1000000	10000000
Serial	0.000738134 s	0.96351 s	10.8233 s
Parallel	0.0402258 s	0.796427 s	8.47063 s

Merge sort

Thread-safe მონაცემთა სტრუქტურები

- std::mutex
- Thread-safe stack:
  - Stack-ის ფუნქციები:
    - Push(), pop(), top(), empty(), size();
- პრობლემები:
  - empty() და size()
  - ელემენტის მნიშვნელობის წაუკითხავად წაშლა

Thread A	Thread B

If (!s.empty())
 	
Int const value = s.top();
 	
s.pop();
do_somthing(value);
	If (!s.empty())
	
	Int const value = s.top();
	
	s.pop();
	do_something(value);

Thread-safe stack-ის დიზაინი

1. Top() და pop() ფუნქციების გაერთიანება
2. pop() ფუნქციას არგუმენტად გადავცეთ ცვლადი, რომელშიც შეინახება stack-ში ბოლოს ჩაწერილი ელემენტი წაშლამდე
3. pop() ფუნქციას დავაბრუნებინოთ პოინტერი stack-ში ბოლოს ჩაწერილ ელემენტზე და შემდეგ წავშალოთ ის.
4. როგორც მე-2, ასევე - მე-3 მეთოდის რეალიზაცია.


THANK YOU
THANK YOU!
THANK YOU


TIME FOR QUESTIONS
12
1
2
3
4