The Algorithm of Parallel Programming Using "Small Delay"

Natela Archvadze[1], Merab Pkhovelishvili[2], Lia Shetsiruli[3], Otar Ioseliani[4]

[1] Department of Computer Sciences Faculty of Exact and Natural Sciences Ivane Javakhishvili Tbilisi State University, Tbilisi, GEORGIA

[2] Department of Programming N.Muskhelishvili Computing Mathematic Institute of Georgian Technical University, Tbilisi, GEORGIA

[3] Department of Mathematics and Computer Science Shota Rustaveli State University, Batumi, Georgia

[4] Georgian-American University,Tbilisi, Georgia

South-Caucasus Computing and Technology Workshop
SCCTW'2016,3-7 October, Tbilisi, Georgia

The algorithms of stable sorting

- *Bubble sort* — If the elements aren't located sequent it's being performed changing of indexes for the pairs of elements .
- *Cocktail sort*.
- *Gnome sort* — Combines *Bubble sort* and *Insertion sort*.
- *Insertion sort* — Makes definition of element's place in sorted list and sets the element there..
- *Merge sort*— Is being performed splitting of list in two parts, sorting of them and then merging with each other.
- *Tree sort*.
- *Timsort* — Combined algorithm, It's being used *Insertion sort* and *Merge sort*.
- *Counting sort*.
- *Bucket sort*.
- *queue sort* (Digital sort).

Processing arrays

(a1 a2 a3 a4 …… an)

OP(a1 a2 a3 a4 …… an)

OP is Σ,Ω, average, max,min… any operation on array elements.

Splitting array in k parts

(k – is the number of cores)

OP((a11 a12 a13 a14 …… a1n/k)

(a21 a22 a23 a24 …… a2n/k)

(a31 a32 a33 a34 …… a3n/k)

…..

(ak1 ak2 ak3 ak4 …… akn/k))

The principal of splitting into sub –arrays

(a1 a2 a3 a4 a5 a6 a7 a8 . . . An)

If the length on n – array and amount of k – cores is known

(a1 a2 a3…an/k) (an/k+1 $a_{n/k+2}$… $a_{2n/k}$)…

If not "The principal of cards distribution"

(a1 a2 a3 a4 a5 a6 a7 a8 . . . An)

(a1 ak+1 …..)

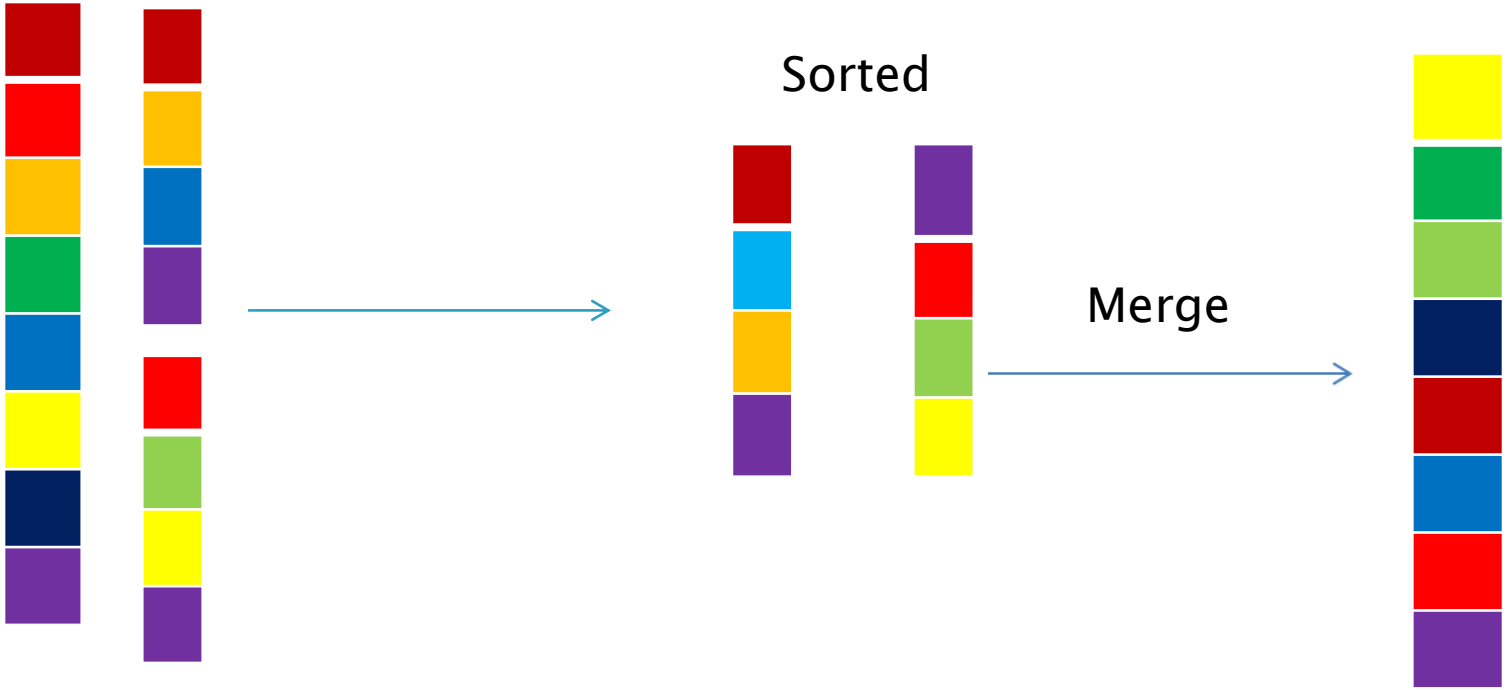
(a2 ak+2 ….)

(a3 ak+2 ….)

…

(ak a2k … ank)

Merge sort



The example of merge

- A 1 3 8 9 10
- B 2 4 5 6 7
- C –
- A 3 8 9 10
- B 2 4 5 6 7
- C 1
- A 3 8 9 10
- B 4 5 6 7
- C 1 2
- A 8 9 10
- B 7
- C 1 2 3 4 5 6
- A –
- B –
- C 1 2 3 4 5 6 7 8 9 10

The principal of new effective sorting algorithm working using parallel programming

- Splitting of initial array is performing by the principal of "distribution of cards".
- First element is being transferred to first core
- The second element to second etc.
- *n*th – (*n* is a number of cores) to *n*th core.
- The element *n+1* is not only transferred to first core, it's being executed the operation of sorting as well with the elements already located there.
- The *n+2* element is being transferred analogically to second core and it's being executed for sorting the elements located there. The process is continuing until all of array elements will not be split into sorted separately *n* subarrays. .
- The traditional merge operation is being executed after.

Conclusion

- The main win of time happens because sorting of arrays starts not after split arrays it starts in parallel with splitting.
- Each core starts sorting operation in parallel with “small delay” in compare with previous core, with received each new element and located there elements of already sorted array.

The principal of splitting into sub arrays

- If we want to sort the array n- of length
- (a1 a2 a3 a4 a5 a6 a7 a8 . . . An)
- According to the principal of “distribution of cards” the elements of
- (a1 a2 a3 a4 a5 a6 a7 a8 . . . An) array will be sent to existed in the order cores.
- As soon as the second element will get on the core it will start searching for its place in sorted elements there.

- (a1 ak+1 …..)
- (a2 ak+2 ….)
- (a3 ak+2 ….)
- …
- (ak a2k … ank)

The parallel algorithm

- (a11 a12 a13 a14 …… a1n/k)
- (a21 a22 a23 a24 …… a2n/k)
- (a31 a32 a33 a34 …… a3n/k)
- <
- …
- (ak1 ak2 ak3 ak4 …… akn/k)
- And then to recursively selected subarray.
- After the sorting of subarrays on cores will be finished, we will use merge sort.

Example

- Let's consider array of 16 numeric elements ascend sorting on computer with 4 cores for better visualization.
- Have given:
- (21 53 11 23 17 42 85 19 57 24 88 99 34 73 87 32)
- On the first step, the first four elements are being sent in order on the 4th core:
- On 1st core 21
- On 2nd core 53
- On 3rd core 11
- On 4th core 23

After the 5th element 17 is compared with 21 and because is less it's being added on first place, similarly with other cores we will get:

- 1 core (17 21)
- 2nd core (42 53)
- 3rd core (11 85)
- 4th core (19 23)
- **at the last core we will get sub arrays:**
- 1st core (17 21 34 57)
- 2nd core (24 42 53 73)
- 3rd core (11 85 87 88)
- 4th core (19 23 32 99).

- Merging being performed by couples
- After merging we will get two subarrays (in parallel)
- (17 21 24 34 42 53 57 73) and

  (11 19 23 32 85 87 88 99)
- Merging of them at the end.

(11 17 19 21 23 24 32 34 42 53 57 73 85 87 88 99)

In algorithm were used

- Splitting using the principal of “distribution of cards”;
- Find the place for element in “Phonebook”;
- Sorting using merge;
- “The Parquet laying parallel task”.

Conclusion

- The major news of given method consists of arrays splitting and sorting in parallel.
- Which provides the possibility to start working with cores with small delay in compare with other cores.
- The win of time is caused because there is not delay before splitting arrays in subarrays.

Literature:

1. Simon Marlow. Parallel and Concurrent Programming in Haskell. 2014.
2. R. Miller, L. Boxer. Algorithms Sequential & Parallel: A Unified Approach. M Binom 2015.
3. N. Archvadze, M. Pkhovelishvili, L.Shetsiruli, O.Ioseliani. Usage of Logic for Parallel Verification of Haskel Programs. “Computer Sciences and Telecommunications”. 2016 | No.3(41). http://gesj.internet-academy.org.ge