**Bucket sort**

all-to-all approach vs parallel approach

Proiect realizat de Vulcănean Ionel

Pentru a scoate în evidență cât mai bine diferențele dintre aceste 2 metode de a sorta un șir de numere cu metoda bucket sort am colectat timpii de comunicare, și cei de computație ai fiecărui algoritm pe mai multe procese și pe 2 seturi de numere, unul de 5 milioane, iar altul de 10 milioane.

**5 milioane de numere**

O imagine care conține text, Font, captură de ecran, număr

Descriere generată automat

În primă parte am colectat timpii de executie și am generat această diagram cu ajutorul valorilor preluate, aceasta scoțând în evidență diferențele de performanță dintre cei 2 algoritmi. Remarcăm următoarele:

* La execuția pe 2,4 procese se remarcă o mică îmbunătățire în abordarea paralelă
* La execuția pe 1, 6-16 procese diferențele sunt nesemnificative, timpii deșii fiind diferiți, luați împreună sunt asemănători
* La execuția pe 20, respectiv 24 procese, se remarcă că timpii in abordarea paralelă încep să crească din ce în ce mai mult

O imagine care conține text, captură de ecran, linie, Interval

Descriere generată automat

În graficul din stânga avem un grafic generat pe baza speed-up-urilor calculați cu ajutorul timpiilor prezentați anterior. În graficul prezentat nu se vede o diferență prea mare între cei 2 algoritmi sirul de elemente este prea mic, iar acest lucru se va vedea in graficele următoare.

**10 milioane de numere**

O imagine care conține text, captură de ecran, număr, Font

Descriere generată automat

La fel ca în prima diagrama, am colectat timpii doar că pentru un set de 10 milioane de numere și pe baza acestora am generat o diagramă. În aceasta remarcăm:

* Exact ca în prima diagramă, la început varianta all-to-all are performanțe mai slabe, pe 2, respectiv 4 procese
* De la 6 la 8 procese se observă o ușoară creștere a timpiilor în varianta paralelă
* În intervalul de 12-24 procese timpul de comunicare creaște din ce în ce mai mult în varianta paralelă. Acest lucru se întâmplă întrucât aceast algoritm se foloseste de functia MPI\_Bcast care distribuie tot șirul de numere către fiecare process, iar acest lucru este foarte costisitor dacă numărul de procese este din ce în cem ai mare
* În același interval și timpul de comunicare din varianta all-to-all crește, dar este o creștere destul de mica și controlată

O imagine care conține text, captură de ecran, linie, Font

Descriere generată automat

La fel ca în cel de-al 2lea grafic am calculat speed-up-ul pe baza timpilor anteriori și am generat graficul din stânga. Se remarcă diferențele enumerate anterior mult mai bine decât în cel de-al 2-lea grafic.

* În prima parte abordarea paralelă are performanțe mai bune decât cea all-to-all
* Ulterior abordarea all-to-all are o performanță din ce în cem ai bună, diferențele fiind vizibile

**Concluzii**

* Abordarea all-to-all are o performanță foarte bună pentru un șir cât mai mare de numere și pe cât mai multe procesoare datorită numărului mic de elemente distribuite.
* Abordarea paralelă are o performanță mai bună pe un număr mic de procesoare întrucât distribuirea șirului de numere devine foarte costisitoare pentru un număr mare de procese.