Furkan Kaya - 191216002 - UNIO92 - Final - 1

Soru 1-)

Elimizdeki kaynakların kullanabildiğimiz kadar kullanıp walldock timemızı azaltmak için paralel programlama önemlidir. 4 core dururken neden tek coreda çalıştıralım ki?

Soru 2-)

Openmpide mpi da paralelleştirme kütüphaneleri/apileri dir.

mpi → message passing interface → mesajlaşma → arayüzü

corelar arasındaki iletişimimizi gerçekleştiren kütüphanedir.

Openmp ise biraz daha farklı çalışır. Shared memorydir haberleşmeyle protokollerle Filan uğraşmayız. Kendisi halleder.

Mpi corelar üzerinden çalışır, openmp threadlar üzerinden çalışır.

mpida coreler sadece kendi bellekllerini görebilir privitedir.

Openmpide threadlar hepsini görebilir sharedtır - private de yapılabilir

Hibrit çalıştırabiliriz. Open mp sanal olarak threadları ocarak çalışır. Distributed bir sistemde çalışcaksak mpi kullanmalıyız shared memory değil çünkü.

~~Soru 3)~~

Soru 4)

run 1 de seri kodumuzun süresi var

run 2 de de openmp kullanmamıza rağmen single thread çalışır ve seriden çok bir farklı olduğu söylenemez

run 3de iş yükü 2 thread için fazla geldi azaltabildiği kadar azalttı süreyi - 2 thread yetmedi

run 4 de // yorum satırına alınmış openmpye bırakmışız istediği gibi takılıyor alabildiğince alıp threadlar, gerektikçe açıyor en optimali bu sistemimizi iyi bilmiyorsak.

Furkan Kaya 191276002 - uni092 - final - 2

Soru 4-2)

run 4-5-6 çok bir fark çıkmamış 64 dü dahada arttırsaydık da çok bişey değişmeyecekti. bir süre sonra threadlara iş yükü kalmıyor boşa açılmış oluyor. en iyisi Open mpi nin kendisine bırakmak.

Soru 3-)

- mpi init başlat
- comsize ve comrank ile değerleri al
- if 0 == myrank veriyi al A matrisine yaz
- mpi scatter ile A matrisini eşit olarak paylaş
- değeri hesapla ve değer değişkenine yaz - hepsinde çalışır -
- mpi Reduce ile değer değişkenini 0. core/mastera mpi_sum ile topla
- if rank == 0 ekrana toplanan Değeri bas
- mpi_finalize ile mpi bitir.

(1)

mpi_init
mpi_comm_size
mpi_comm_rank

(2)

mpi_scatter

(3)

çişlemi

(4)

mpi_Reduce
mpi_finalize