Gökçe DEMİR
141044067

**SİSTEM PROGRAMLAMA FİNAL PROJESİ RAPORU**

Parellel programlamayı öğretmeyi amaçlayan bu ödev de server ve clients isimli iki ayrı program yazmamız beklendi. Bu iki program arasındaki haberleşmeyi socket aracığıyla sağlıyorum. Clients programım parametre olarak aldığı client sayısı kadar thread oluşturur, her bir thread socket aracığıyla row ve columnu server'a iletir. Bu esnada senkronizasyonu sağlamak için semafor kullandım. Server her bir client için matris generate eder. Aynı zamanda solve ve verify eden generate ettiği bu matrisi. Bu kısımda da server da 3 tane fork yapıp birbirlerini wait yapmadan aynı anda 3 işlemi de gerceklestiren 3 process olusturdum. Bu işlemlerin aynı anda çalışıp senkron bir şekilde ilerlemesi içinde semafor kullanma ihtiyacı duydum. Bu üç process birbiri ile matris alışverişini sağlayabilmesi için shared memory oluşutuyorlar. Generate ve Solve fonksiyonları arasındaki shared memorye generate edilen matrisleri yazıyorum. Aynı şekilde solve ve verify arasındaki shared memory de ise çözüm vektörü olan x'i aktarıyorum. Generate edilen A ve B matrisleri solve fonksiyonu için de A.x= B lineer denklemini çözmeye çalışır. Bu kısımda bizden 3 ayrı yöntemle bu denklemi çözmemiz bekleniyordu. Uzun araştırmalarıma rağmen QR factorizition ve SVD yöntemi ile alakalı pek bir konu anlatımı ve örnek bulamadığımdan sadece pseudo-inverse metodunu kullanarak solve işlemini gercekleştirebiliyorum. Bu yöntemin de şöyle bir şartı var matrisin row > col 'undan büyük olduğu sürece bu işlemi gerçekleştirebiliyor. Verify fonksiyonum da error vektörünü ve error normu hesaplar. Error normu socket aracığıyla cleinta iletilir. Her bir client kendi logu oluşturur ve socket aracığıyla eriştiği A, B ve error normunu bu loglara yazar. Ayrıca clients programı tek bir tane log oluşturur ve bu log dosyasında da her bir clienttın serveri baglı kaldıgı zamanın ortalaması ve standart sapması bulunur. Server her servis ettiği client için log oluşturur ve bu loglarda A , B ve çözüm vektörü olan x bulunur. Aynı zamanda ekrana o anda o bağlı olan clientların sayısını basar.

Program genel akışı dışında bizden istenen Ctrl-C'yi handle etmiş olup servera ctrl-c geldiği anda çalışır olan clientlarında kapanmasını sağladım. Bunu tüm clientsların idsini bi dosyaya yazıp ölen clientin id'sini silerek geri kalanların kill komutu ile öldürülmesini sağladım. Eğer server çalışmadan client çalışır ise hata verip çıkıyor. Bunu da serverin calısır calısmaz olusturdugu bir dosya ile sağlıyorum. Clients programı calıstıgında bu dosyayı bulamazsa serverin calıstırılmamış oldugunu anlayıp cıkıyor. Eğer clients programına ctrl-c gelirse ise sadece kendi ölüyor. Benden istenen thread pool ve yukarda bahsettiğim iki solve metodunu ne yazık ki zaman yetmediğinden yetiştiremedim. Lakin bu

Gökçe DEMİR
141044067

final projesinin paralel programlama mantığını öğretip. Iterative düşünmeden uzak bir şekilde parelellizmi kavrattığını söylebilirim. Tüm dönem boyunca öğrendiğimiz konuların tek bir proje içinde karşımıza cıkıyor oluş öğretici olma açısından yararlıydı.