1.MPI(Message Passing Interface)

MPI (Message Passing Interface) bir bilgisayar iletişim protokolüdür. Dağıtık bellekli bir sistemde paralel program koşan düğümlerin arasındaki iletişim için kullanılan fiilen standart bir protokoldür. MPI uygulamaları Fortran, C, C++ ve Ada programlarından çağrılan kütüphane yordamlarından oluşur. MPI 'ın diğer eski mesaj geçirmeli kütüphanelere olan üstünlüğü; taşınabilir (MPI pek çok dağıtık bellekli mimari üzerinde uygulanmıştır) ve hızlı olmasıdır.

MPI, temel olarak süper bilgisayar üzerinde yürütülen bir işlemin tüm bilgisayarlara parçalanarak ayrı ayrı olarak dağıtılması için kullanılan mesaj gönderme ara yüzüdür [6,7]. MPI, C'de bir fonksiyon, Fortran'da bir alt rutindir ve işlemciler arasında haberleşmeyi sağlamak amacıyla kullanılır. Tüm işlem süreci boyunca işlemciler aralarında haberleşirler. Bu nedenle, MPI kütüphanesi bu haberleşmeleri sağlayacak fonksiyon ve alt rutinleri de içermektedir. İlk MPI çalışmalarına Kasım 1992'de resmi olarak ilk sunum yapılması ile başlanmıştır.[6,7] MPI standardı yaklaşık elli kişilik bir grup tarafından birçok farklı organizasyonda tartışılarak geliştirilmiştir. 1993 yılında süper bilgisayarlar için geliştirilen ilk programların taslakları sunulmuş ve bir yıl sonra ilk MPI standardı olan MPI-1.0 bir çok farklı platformda denerek tamamlanmıştır.[8,9] Bu standart, Fortran 77 ve C'de çağrılacak alt rutin ve fonksiyon isimlerini ve sonuçlarını içermektedir. Nisan 1995'de MPI-2 çalışmalarına başlamıştır. Haziran 1995'de MPI-1.1, Temmuz 1997'de MPI-1.2 ve 1997'nin sonlarında ise, MPI-2 sunulmuştur. MPI-2, MPI-1.1'den faklı olarak paralel I/O işlemleri için paketleri, Fortran 90 dosyalarını ve dinamik işlem yöneticisi gibi ek özellikleri içermektedir.

2.GENEL MPI PROGRAM YAPISI

Bir C programı düşünüldüğünde MPI fonksiyonlarının yer aldığı bir programın yapısı aşağıdaki gibidir[1].

1. Header dosyalarının eklenmesi
2. MPI veri türlerinin belirlenmesi
3. MPI'ın başlatılması
4. MPI haberleşmelerinin yapılması
5. Program içinde yer alan gerekli hesaplamaların yapılması
6. MPI'ın kapatılması
7. Programın sonlandırılması

MPI'da C fonksiyonlarının tümü MPI_Xxxxx şeklindedir. MPI fonksiyonlarının kullanılabilmesi için C kodlarına başına #include ya da #include "mpi.h" şeklinde MPI header eklenmelidir. Başlangıç MPI fonksiyonu, MPI_Init(&argc,&argv) fonksiyonudur ve bu fonksiyon MPI programın çalışıp çalışmadığını kontrol eder. Prosesler arası haberleşmeler, MPI_Comm_size (comm,&size) ve MPI_Comm_rank(comm,&rank) fonksiyonları ile sağlanır. Birinci fonksiyon, sistemde bulunan işlemci (node) sayısını verir. İkinci fonksiyon ise, işlemcilerin 0'dan N'e kadar sayısal olarak sıralamasını sağlar[2,3]. Bu fonksiyon kullanılmazsa, bütün işlemciler 0 ya da -1 değerini alır. MPI'ın sonlandırılması, MPI_Finalize( ) fonksiyonu ile sağlanır. Bu nedenle bu fonksiyon kodun en sonuna eklenmelidir.

3.MPI İLE HABERLEŞME

MPI bir haberleşme protokolüdür ve MPI fonksiyonları temel olarak node'ların aralarında haberleşmelerini sağlar. MPI ile iki temel haberleşme şekli vardır[1]. Birincisi, herhangi bir node'un bir mesajı göndermesi ve diğer node'un ya da node'ların bu mesajı alması şeklinde gerçekleşen haberleşme türüdür. İkincisi ise, node'lar arasında verilerin gönderilmesini ve alınmasını içeren haberleşme türüdür.

3.1. Point-to-Point Haberleşme

Point-to-point haberleşme bir node'un herhangi bir mesajı diğer node ya da node'lara gönderilmesi ve bu mesajın tüm node ya da node'lar tarafından alınması sürecini kapsar. Point-to-point haberleşmesi, bir node'un bir mesajı gönderip diğer node'ların bu mesajı alması prensibine göre çalıştığından işlem süresince senkronizasyon yoktur[2,3]. Birinci node başlangıç olarak mesaj gönderir. Daha sonra ikinci node mesajı alarak işlemi sürdürür. Yani point-topoint haberleşmesi mesaj gönderme ve alma şeklinde iki yönlü işler. Bu haberleşme boyunca her iki node'un olaya katılması gerekir. Aksi takdirde veri transferi gerçekleşemez[5]. MPI'da mesajlar, mesaj zarfı ve mesaj gövdesi olmak üzere iki kısımdan oluşur. MPI mesaj zarfı, üzerinde gideceği hedefin adresinin, dönüş adresinin, göndermek ve alınmak için gerekli diğer bilgilerin yer aldığı günlük hayatımızda kullandığımız mektup zarfına benzer. MPI mesaj zarfı, gönderilecek işlemciyi gösteren kaynak, alıcı işlemciyi gösteren hedef, kaynak ve hedef işlemcilere ait özel haberleşme özelliklerini içeren haberleştirici ve mesajı etiketlendirmek amacıyla kullanılan işaretçi olmak üzere toplam dört kısımdan oluşur. Mesaj gövdesi ise, mesaj verisi, mesaj veri tipi ve mesaj veri sayısı olmak üzere üç kısımdan oluşur[7]. Point-to-point haberleşmesinde, mesaj gönderilmesi "send" argümanıyla, alınması ise "recieve" argümanıyla gerçekleştirilir.

SEND Parametresi

MPI_Send fonksiyonu, istenilen bir veriyi herhangi bir işlemciye göndermek amacıyla kullanılır. Bu fonksiyonla tek bir değişken gönderilebileceği gibi bir dizi ya da bir dizinin herhangi bir kısmı da gönderilebilmektedir. Bir mesaj MPI_Send fonksiyonu kullanılarak gönderildiğinde, mesaj ilk olarak hafızaya alınır ve bloklama fonksiyonları kullanılmamış ise, senkronize olmaksızın hedef işlemciye gönderilir. Gönderilen mesaj mevcut hafızadan daha fazla yer kaplıyorsa, hafıza alanı kullanılamaz ve mesajı göndermekte olan işlemci yeterli hafıza alanı sağlana kadar mesajı bloklar. MPI_SEND parametresinin yapısı aşağıdaki gibidir.

MPI_Send(&input_veri,input_veri_sayısı, input_veri_tipi,hedef_node'un_rankı, işaret_sayisi,MPI_COM_WORLD)

RECIEVE Parametresi

MPI _Recv fonksiyonu, node'lar tarafından mesajın gönderildiğini ve bu mesajın kullanıma hazır olduğunun belirtilmesi amacıyla kullanılır. Bu amaçla MPI_Recv fonksiyonu, MPI_Send fonksiyonundan faklı olarak status şeklinde ekstradan bir argüman içerir. Bir veri hedef noda ulaştığında, mesajı işaretine, kaynağına ve alınan veri sayısına status argümanı yardımı ile ulaşılabilir. Eğer alınan mesaj gönderilen mesajdan fazla ise, mesaj gönderme alma sürecinde hata oluştuğu anlamına gelir. Yani her durumda alınan mesaj ile gönderilen mesaj miktarı aynı olmalıdır. Ayrıca dikkat edilmesi gereken diğer önemli bir nokta ise, gönderilen ve alınan mesaj veri tiplerinin aynı olmasıdır. MPI_Recv parametresinin yapısı aşağıdaki gibidir
. MPI_Recv(&output_veri,output_veri_sayısı, output_veri_tipi,kaynak_node'un_rankı, işaret_sayısı,MPI_COM_WORLD,&status)

3.2. Kolektif Haberleşme Kolektif haberleşme,

bir grup node arasında verilerin gönderilmesini ve alınmasını kapsar. Genel olarak düşünüldüğünde, kolektif haberleşme rutinleri kullanılmaksızın yapılacak tüm işlemler MPI_Send ve MPI_Recv fonksiyonları kullanılarak yapılabilir. Fakat kolektif haberleşme fonksiyonları, bu işlemlerin çok daha kısa bir şekilde yapılmasını sağlar. Ayrıca point-to-point haberleşmeden farklı olarak kolektif haberleşme fonksiyonları yardımıyla, bir node'un birden fazla node'a (one-to-several) ya da diğer node'ların herhangi bir node'a (several-to-one) veri transferi yapmasına olanak sağlanır[8,9]. Kolektif haberleşmenin kullanılması ile program içinde yazılacak MPI kodlarında önemli ölçüde bir azalma olur. Bu ise, program yazılması sırasında yapılacak hatalarının azalması anlamında gelir. Ayrıca kolektif haberleşme rutinleri kullanılması ile node'lar arası yapılan haberleşmeler azaldığından çalıştırılan programın daha kısa sürede tamamlanması sağlanabilmektedir.

Barrier Senkronize Edici Fonksiyon:

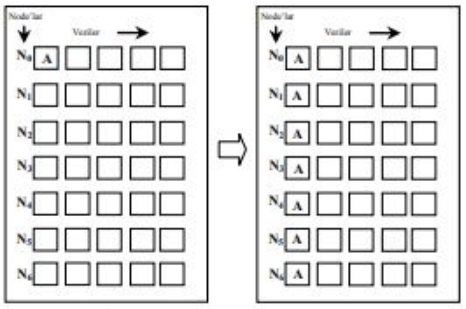
Bu fonksiyon, node'lar arasındaki senkronizeliği sağlamak amacıyla kullanılır ve her bir node herhangi bir işlem yapmadan önce diğer node'ların belirlenen sürece kadar beklemesini sağlar.

MPI_Barrier fonksiyonunun yapısı aşağıdaki gibidir.

……
……
MPI_Barrier (MPI_COMM_WORLD)
……
……

Broadcast Fonksiyonu:

Şekil 1'de görüldüğü gibi broadcast fonksiyonu ile ana node'daki istenilen bir veri, haberleşme grubu içinde yer alan diğer tüm node'lara dağıtılır.



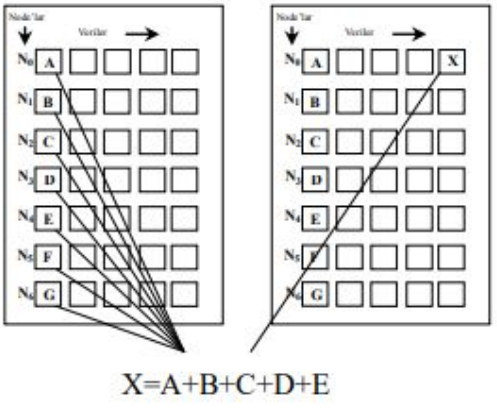
Şekil 1. Broadcast fonksiyon örneği.

MPI_Bcast fonksiyonunun yapısı aşağıdaki gibidir.

……
……
MPI_Bcast (&input_veri,input_veri_sayısı, input_veri_tipi,ana_node'un_rankı, MPI_COMM_WORLD)
……
……

Reduction Fonksiyonu

Reduction fonksiyonu, her bir node'dan istenilen verilerin alınarak toplama, çarpma, minimum ya da maksimum değeri bulma gibi işlemleri gerçekleştirerek ana node'da saklanmasını sağlar[6]. Şekil 2'de verilen reduction fonksiyonu tüm node'lardan aldığı verileri toplayarak ana node'da aktarmaktadır



Şekil 2. Reduction fonksiyon örneği.

Çizelge 1'de Reduction fonksiyonu içinde kullanılan operatörler verilmiştir. Bu

operatörler yardımı ile node'lardan gelen veriler basit bir şekilde aşağıdır.

Operatör	Açıklaması
MPI_MAX	Maksimum Değer
MPI_MIN	Minimum Değer
MPI_SUM	Toplama işlemi
MPI_PROD	Çarpma işlemi
MPI_LAND	Mantıksal AND
MPI_LOR	Mantıksal OR

Çizelge 1. Reduction fonksiyonunda kullanılan bazı operatörler.

KAYNAKLAR

[1] http://www.beowulf.org

[2] http://www.top500.org/

[3] Kaleci, D., Kaya O. A., Karakaplan, M., Şahin A., 'The Linear Acoustic Field Calculation on 15 node Linux Cluster', Balkan Physics Letters, 2008 Special Issue, Boğaziçi University Press, ISSN 1301-8329, 2008, p.215-219.

[4] Uehara, H., Tamura M., Yokokawa M., 'An MPI Benchmark Program Library and Its Application to the Earth Simulator', Lecture Notes in Computer Science, vol.2327, January 2002,pp 351-356.

[5] Stefano Cozzini, Axel Kohlmeyer, and Roger Rousseau, 'Benchmark Analysis of 64-bit Servers for Linux Clusters for Application in Molecular Modeling and Atomistic Simulations', Proceedings of the 7th LCI International Conference on Clusters, 2006.

[6] http://www-unix.mcs.anl.gov/mpi/

[7] http://lam-mpi.org

[8] http://www.lanl.gov/roadrunner/

[9] http://domino.research.ibm.com/comm/research_project s.nsf/pages/bluegene.index.html