GAZİ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BM-311 Bilgisayar Mimarisi

ELİF GİZEM ZEDEF

191180093

Araştırma Ödevi

Cache Coherence İçin Kullanılan Protokoller

İÇİNDEKİLER

ŞEKİLLER

1 GİRİŞ

Bu raporda multicore ve çok işlemcili sistemlerde önbellek tutarlılığı için kullanılan protokoller incelenmiş ve açıklanmıştır. Bunlar snoopy tabanlı ve dizin tabanlı olmak üzere iki ana başlıkta incelenmiştir. MSI, MOSI, MESI protokolleri açıklanmıştır.

Protokoller anlatılmadan önce hafıza tutarsızlığı problemi önbellek ve önbellek tutarsızlığı problemi kavramlarının açıklanması gerekmektedir. Konun daha iyi anlaşılması açısından problem açıklandıktan sonra çözümler ele alınmıştır.

Başlıklar sırasıyla hafıza tutarlılığı problemi, önbellek, önbellek tutarlılığı problemi, önbellek tutarlılığı protokolleri ve sonuç olarak ayrılmıştır.

2 HAFIZA TUTARLILIĞI PROBLEMİ

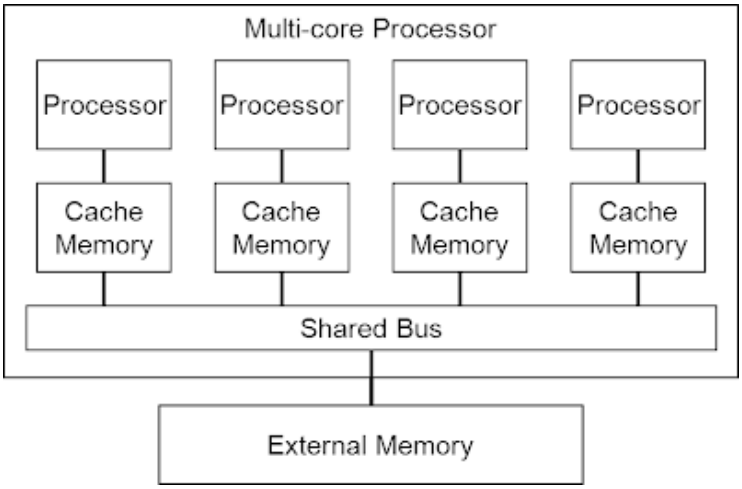
Von Neumann mimari modelinde komutlar programcı veya derleyicinin belirlediği sırada çalıştırılmaktadır. Bu modelin en önemli özelliklerinden birisi işlem sırasında ihtiyaç duyulan veriyi ana bellekten bloklar halinde almasıdır. Tek işlemcili ve tek çekirdekli sistemlerde büyük verim sağlamaktadır. Fakat bunun yanı sıra çok çekirdekli veya çok işlemcili sistemlerde çekilen blokların içerisinde başka bir çekirdeğin veya işlemcinin sonucu olabileceğinden veya onun da ihtiyacı olabileceğinden sorunlar oluşmaktadır. [1,3,8]

Çok çekirdekli veya işlemcili sistemlerde ana hafızaya erişim için belirli protokoller belirlenmiştir. Amaç veri kirliliğini önlemektir. Aynı zamanda yanlış bilgi sonucu çalıştırılan uygulamadan programcının istediğinin dışında bir sonucun elde edilmesinin önüne geçilmek istenmiştir. Çok çekirdekli ve çok işlemcili sistemlerde hafıza tutarlılığı sorunu ile hafıza tutarlılığının bir parçası olan önbellek tutarlılığı sorunu da oluşmaktadır. [1,7]

3 ÖNBELLEK

Bir işlemcinin kullanacağı bir veriyi doğrudan ana hafızadan alması çok uzun sürelere mal olabilmektedir. Bu süreyi kısaltmak için önbellekler kullanılmaktadır. Bir işlemci, kullanacağı veriyi önce registerlarda orada bulamazsa önbellekte orada da bulamazsa ana hafızada aramaktadır. Bu hiyerarşi veriye mümkün olduğunca kısa sürede ulaşılması içindir. [2]

Registerlarda ve önbellekte bulunmayan veri ana bellekten önbelleğe bloklar halinde alınmaktadır. Önbellekten, registerlara ise tek bir satır olarak iletilmektedir. Ana bellekten verilerin bloklar halinde alınmasının sebebi kullanılacak olan veriye yakın olan verilerin, yakın zamanda kullanılma olasılığının bulunmasıdır. Verilerin bloklar halinde alınması performansı arttırmaktadır. [2]



Şekil 3-1 Multicore İşlemci Şeması

4 ÖNBELLEK TUTARLILIK PROBLEMİ

Multicore işlemcilerde önbellek önemli bir rol oynamaktadır. Ana hafızadan veri çekerken geçen zamanı büyük oranda düşüren önbellekler bilgisayarların çok daha hızlanmasına yardımcı olmaktadır. Fakat bu hiyerarşi yararları ile bazı tutarlılık sorunlarını da beraberinde getirmektedir. Mesela C1 ve C2 işlemcinin iki ayrı çekirdeği ve kendilerine ait gizli önbellekleri olsun. Aynı anda B1 adını verdiğimiz bloğu önbelleklerine çeksinler. C1, B1 bloğunu güncellediğinde C2 çekirdeğindeki veriler ile B1 bloğunda bulunan değerler tutarsız olur. Eğer C2 çekirdeği B1 bloğunu tekrar kullanmayacaksa bu durum bir sorun oluşturmamaktadır. Ama tekrar ihtiyaç duyması durumunda yanlış veri alma işlemini önlemek için önbellek tutarlılık mekanizmalarına ihtiyaç vardır. [1]

Genel bir yaklaşım olarak bloklar kendileri için birer izin değeri tutarlar. Bu değerler o bloktan veriyi isteyen işlemcinin hangi işlemlere izninin olduğunu göstermektedir. Örnek vermek gerekirse bir bloğu bir çekirdek okuma ve yazma işi ile tutarken diğerleri sadece okuma yazmak için tutabilmekteler. Tutarlılık protokolleri bu değerlerin doğru atandığından emin olmak için kullanılmaktadır. Bu işlemler için çeşitli durumlar bulunmaktadır. Bunlar aşağıdaki tabloda gösterilmiştir. [1]

Durum	Verilen İzin	Tanım
Değiştirilmiş (M)	Okuma, Yazma	Diğer çekirdekler I veya NP durumundalardır.
Sahipli (O)	Okuma	Diğer çekirdekler S, I veya NP durumundalardır.
Özel (E)	Okuma, Yazma	Diğer çekirdekler I veya NP durumundalardır.
Paylaşılmış (S)	Okuma	Başka hiçbir çekirdek M veya E durumunda değildir.
Geçersiz (I)	Yok	Yok
Mevcut Değil (NP)	Yok	Yok

Şekil 4-1 Önbellek Durumları Tablosu

5 ÖNBELLEK TUTARLILIK PROTOKOLLERİ

Bir veri birden fazla önbellekte eşzamanlı bulunabilmektedir. Bu eşzamanlılık ile gelen tutarsızlık probleminin belirli iki ana yöntemi vardır. Snoopy tabanlı önbellek tutarlılık protokolü ve Dizin tabanlı önbellek tutarlılık protokolü olarak alt başlıklarda detaylı bir şekilde açıklanmıştır.

Bu başlığın altında snoopy ve dizin tabanlı önbellek tutarlılık protokolleri açıklanmıştır.

5.1 Snoopy Tabanlı Önbellek Tutarlılık Protokolleri

Snoopy tabanlı önbellek protokolü tek seferde sadece ama sadece tek bir işlemcinin yazmasına izin verilmektedir. Bu protokolde her çekirdek o bloğun bir kopyasına sahip olup olmadıklarını kontrol etmekte ve buna göre yanıt vermektedir. İşlemlerin düzgün gerçekleştirilmesi için ayrıca istek sıralamasının da tutulması gerekmektedir. Bu nedenle genellikle veri yolu tabanlı bir yapıya sahiptir. [1,3,4]

Snoopy önbellek ile tek işlemcili önbellek arasındaki temel farklar önbellekteki bilgileri depolayan önbellek denetleyicisi de durum geçmişine göre önbellek tutarlılığı kontörlü yapan sonlu veri yolu denetleyicisidir. Bu protokol basit ve uygulanması kolaydır. Sequent Computer Systems'in Simetri çoklu işlemcisi ve Alliant Computer sistemlerinin önbellek tutarlılığını korumak için yazma-geçersiz kılma ilkelerini kullanan Alliant FX gibi birçok ticari, veri yolu tabanlı çok işlemcili bu protokolü kullanmıştır. [3,7]

Önbellekler ve ana hafıza ile iletişimin veri yolu (bus yapıları) ile yapılıyor olması ve çoklu işlemcilerin artması ile verilerin hafızadan çekilme hızı ihtiyaca nazaran çok daha yavaş kalmıştır. Bu durum da önbellek ve hafıza arasında bir darboğaz oluşmasına sebep olmuştur. Veri yollarının bant genişliğini arttırmak için ek olarak yeni veri yolları eklenebilmektedir. Ama bu da fiziksel olarak maliyetli ve bir yerden sonra yapılması zor bir hal almaktadır. [3,7,8]

Bu protokolün çoklu sistemlerde kullanılması elverişli değildir. Her yazma isteğinde diğer belleklerle iletişim kurması büyük bir dezavantajdır. [3,7,8]

5.1.1 MSI Protokol

MSI protokolü basit düzey bir protokoldür. Her harf önbelleğin alabileceği durumu sembolize etmektedir. Her blok aşağıdaki gibi durumlara sahip olabilmektedir. [4,5]

Modified : Başka işlemciler tarafından değiştirilmiş

Shared : Değiştirilmemiş konumda ama en az bir işlemci ile paylaşılmış

Invalid : Blok geçersiz. İhtiyaç haline tekrar fetch edilmesi gerekmektedir.

5.1.2 MESI Protokol

MSI protokolü gibidir. MSI'dan farklı olarak exclusive değerinin gösterilmesidir. Bu da alınmış olan bloğun sadece o önbellekte bulunduğunu ve ana bellek ile örtüştüğünü göstermektedir. [4,5,6]

5.1.3 MOSI Protokol

MSI protokolü gibidir. MSI'dan farklı olarak owned değerine sahiptir. Bu da mevcut işlemcinin bu bloğa sahip olduğunu ve blok için diğer işlemcilerden gelen istekleri yerine getireceğini belirtmektedir. [4,5,6]

5.2 Dizin Tabanlı Önbellek Tutarlılık Protokolleri

Bu protokol üzerinde bir veri yolu şart değildir. Bu protokolde birden fazla yazma işlemine izin verilmektedir. Bunun kontrolü ise her yapılan değişiklikte o bloğa sahip olan işlemcilere de yeni verinin gönderilmesi ile sağlanmaktadır. Bir bloğu aynı anda birden fazla işlemci kullanabileceğinden dizin tabanlı protokolde hangi işlemcinin hangi bloğu aldığının bilgisi tutulmaktadır. İşlemler sırasında işlemcinin hafızadan veriyi yüklemeden önce izin istemesi gerekmektedir. Alınan veri bloğunda bir değişim yapılacağı sırada ya yapılmadan önce ya da yapıldıktan sonra bunun hafızaya bildirilmesi gerekmektedir. Sonrasında da bu bloğa sahip belleklerin güncellenmesi gerekmektedir. [3,7,10]

6 SONUÇ

Çok işlemcili veya çok çekirdekli sistemlerde paralel çalışmadan kaynaklı olarak aynı anda kullanılan verilerden kaynaklı veri tutarsızlığı ortaya çıkmaktadır. Bu problemi ortadan kaldırmak adına protokoller üretilmiştir. Bunları snoopy ve dizin tabanlı olmak üzere iki ana başlıkta incelenmiştir. Snoopy tabanlı protokol veri yollarını gözetler iken dizin tabanlı protokol hangi bloğun hangi işlemde olduğun tutarak veri tutarsızlığını gidermeyi amaçlamışlardır. Önemli olan Protokollerden MSI, MESI ve MOSI protokolleri de açıklanmıştır.

MSI, MESI, MOSI protokollerinin ortak olan özelliklerinden birisi M,S ve I önbellek durumlarına izin veriyor olmalarıdır. Birbirlerinden farklı olarak E veya O gibi başka durumları da vardır. Bu durumlar yukarıda açıklanmıştır.

Snoopy tabanlı protokoller günümüz sistemlerinde kullanılmaya uygun görülmemektedir. Kullanılması fazla maliyetli ve uğraşlıdır.

7 KAYNAKÇA

[1] Soltaniyeh, M. R. (2015). *Boosting performance of directory-based cache coherence protocols by detecting private memory blocks at subpage granularity and using a low cost on-chip page table* (Doctoral dissertation, Bilkent Universitesi (Turkey)).

[2] Atamaner, M. (2020). *Çok çekirdekli görev-kritik işlemciler için önbellek tasarımı ve gerçeklenmesi* (Master's thesis, TOBB ETÜ Fen Bilimleri Enstitüsü). [3] Cache Coherence Protocols in Shared-Memory Multiprocessors

[4] Bigelow, L., Narasiman, V., & Suleman, A. An Evaluation of Cache Coherence Protocols.

[5] Tsaliagos, D. (2011). *Design and Implementation of a Directory based Cache Coherence Protocol*. Technical Report FORTH-ICS/TR-418 May.

[6] Shukur, H., Zeebaree, S., Zebari, R., Ahmed, O., Haji, L., & Abdulqader, D. (2020). Cache coherence protocols in distributed systems. *Journal of Applied Science and Technology Trends*, *1*(3), 92-97.

[7] Lian, X., Ning, X., Xie, M., & Yu, F. (2015, July). Cache coherence protocols in shared-memory multiprocessors. In *2015 International Conference on Computational Science and Engineering* (pp. 286-289). Atlantis Press.

[8] Mutlu, O., *Computer Architecture: Cache Coherence*. Carnegie Mellon University (Ders Kaynağı)

[9] Nakshatra, S., *Cache Coherence Protocols* (Computer Architecture(EECC551) Dersi kaynağı)

[10] Arora, H., Mukherjee, R., Bej, A., & Adak, H. (2014, September). Directory based cache coherence modeller in multiprocessors: Medium insight. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 2611-2617). IEEE.