CACHE COHERENCE

ALİ HADİ ALTUNGÖK

GAZİ ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ
BİLGİSAAR MÜHENDİSLİĞİ
BİLGİSAYAR MİMARİSİ
Prof.Dr. M.ALİ AKÇAYOL

ARALIK 2020

ÖZET

Cache Coherence, Multicore ve çok işlemcili sistemlerde işlemcilerin cache belleklerinin ortak Main Memory (RAM) i kullanması sonucu, aynı veriyi barındırıyor olabileceği ve bu verinin o cache belleklerdeki mevcut işlemciler tarafından birbirinden habersiz bir şekilde işlenmesi durumunda, veriler arasında kesin bir tutarlılık olması zorunluluğu durumudur. İşlemcinin yüksek kapasitesinin verimli bir şekilde kullanılması amacı ile geliştirilen cache bellekli işlemcilerin multicore ve çok işlemcili sistemlerde kullanılması durumunda veri tutarlılığını sağlamak için her işlem yapıldığında Main Memorye geri gidip veriyi yazmak yerine zaman tasarrufu ile verilerin cacahe bellekten atılacağı zamana kadar Main Memorye götürülmediği durumlar geliştirilmiştir. Bu dumlarda da veri tutarlılığının sağlanması için cache belleklerdeki veriler işlenirken bazı protokoller geliştirilmiştir. Bu araştırma ödevinde Multicore ve çok işlemcili sistemlerde ki cache belleklerden veriler alınıp işlenirken sağlaması gereken Cache Coherence (Önbellek tutarlılığı) durumunu ayrıntılı bir şekilde açıklayıp, bu durumda kulanılan protokolleri, çeşitlerini, nasıl kullanıldığını, bu protokollerin mantığını ve algoritmik yapısını detaylı bir şekilde inceleyeceğiz. Araştırmayı yaptığım kaynakları referans göstererek Multicore ve çok işlemcili sistemlerde Cache Coherence için kullanılan protokoller hakkında kapsamlı çalışma ödevini gerçekleştirmiş olacağız.

CACHE COHERENCE

Cache Coherence, Multicore (çok işlemcili) sistemlerde her işlemcinin Cache belleklerinde (Ön bellek) bulunan verilerin ortak olması durumunda farklılık göstermemesi, tutarlı olması gerektiği ve bunun bazı protokollerle sağlanması sonucu oluşan durumdur.

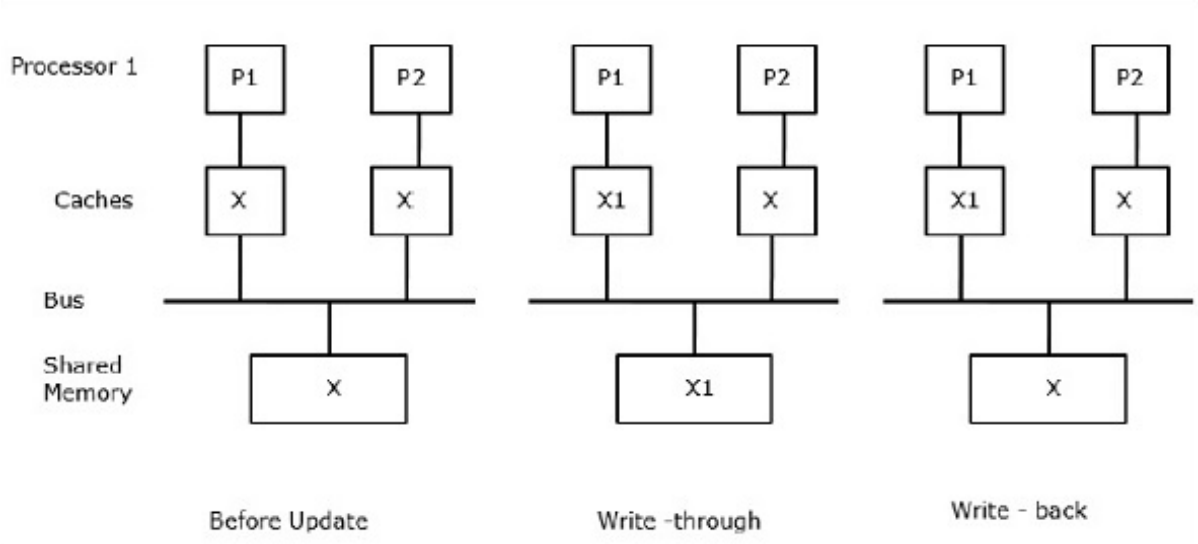
Cahche bellek(ön bellek) işlemcinin data erişim sürecini kısaltmak için kullanılan Main mamoryden yani RAM den data ve komut verilerini alıp daha kısa bi şekilde işlemciye iletilmesi için üretilmiş diğer hafıza birimlerinden çok daha küçük olan hafıza birimidir. İşlemcinin birim zamanda iş yapma kapasitesi çok yüksek olduğundan, hafıza birimlerinden verinin işlemciye aktarılma süreci, uzun bir zaman alıp işlemcinin yüksek kapasitesinin verimli bir şekilde kullanılmasını sınırlandırması sonucu geliştirilmiş hafıza birimleridir.

Main memorydeki dataların bloklara ayrılması ile cache belleğin her bir satırı bu bloklardan bir tanesini alabilmektedir. Hafızadaki bir bloğun cache bellğe nasıl yerleşeceği ise maping fonksiyonları ile gerçekleştirilir. Cache belleğe verilerin getirilip yazılması ise replacement fonksiyonları ile gerçekleştirilir. Bu fonksiyonlar ile tekrar tekrar kulanılan veriler cache bellekte tutulmuş olur ve böylece bu veriye ulaşmak için tekrar main memoriye gidilerek zaman kaybedilmemiş olur. Data, direk cache belleğe bakılarak orada ise veri okunarak işlmler gerçekleştirilmiş olur. Verinin orada olmadığı durumlarda main memorye gidilerek cache bellğe getirilir. Böylece tekralı kullanılan veriler RAM e geri gönderilmemiş olup, işlemcinin kapasitesinin verimli kullanılması sağlanmış olur.

Multicore işlemcili çok işlemcili sistemlerde ise main memorydeki datalar işlemciler tarafından ortak bir şekilde kullanılmaktadır. Her işlemci kendisine ait cache belleklere sahiptir. Bu durumda cache belleklerin ortak datayı alması ve aynı data üzerinde işlem yapıyor olması gibi bir durumla karşılaşılabilir. Bu durmlarda ise veri tutarlılığı bozulmaması gerekmektedir. Aynı veriyi işlemcilerden biri RAM den alıp işlemek istediğinden o veri başka işlemcinin cache belleğinde bulunuyor, değişmiş ve RAM e yazılmamış olduğu durumlarda eski olan yani güncel olmayan verinin alınıp işlenmesi veri tutarsızlığı oluşturacaktır.

Verilerin cache bellekler arasındaki tutartlılığına da Cache Coherence (Önbellek tutarlılığı) denir. Veri tutarsızlığını önlemek için geliştirilmniş olan Cache Coherencede kullanılan birçok protoköl bulunmaktadır. Bu protokoller bir verinin aynı anda birden fazla işlemcide işlenirken tutarsızlaşmasını önler.

Ön bellekteki veriler değişmiş ise bu verilerin hafızaya (main memory) aktarılması gerekmektedir. Verilerin değişmesi durumunda bu verilerin hafızaya aktarılması çok önemlidir. Verinin hafızaya yazılması şu şekilerde gerçekleştirilebilir; veri değişir değişmez hafızaya yazılır(Write through), veri değişir değişmez değil ön bellekten atılacağı zaman hafızaya ayzılır(Write back), veri ilk değiştiğinde hafızaya yazılmakta fakat sonraki değişimlerde hafızaya yazılmamakta en son cache bellekten atılacağı zaman hafızaya yazılmaktadır(Write once). Multicore ilşlemcilerde veri değiştirildiğinde Write through yöntemi ile aktarılıyorsa veri tutarlılığı konusunda sorun olmayacaktır fakat bu durumda Bus trafiği yoğun olacaktır. Write back yada Write once yöntemleri ile yazıldığı durumlarda, Çok işlemcili sistemlerde veri tutarsızlığı gerçekleşetirebilecek durumlar olabilecektir. Bu durumu aşşağıdaki şema üzerinden gözlemleyebiliriz.



Bazı kaynaklarda donanımsal ve yazılımsal protokollerden bahsedilmektedir. Yazılımsal protokolde cache belleğe sadece belirli yapıdaki datalar gönderilmektedir. Hangi verinin verilip verilmeyeceğini derleyici belirler. Bu datalar sadece okunabilir, readonly verilerdir. Bu durumda da cach bellekteki bu veriler sadece okuma işlemine tabi tutulacağından, data üzerinde bir değişiklik gerçekleşmeyeceğinden ötürü veri tutarsızlığı durumu ile karşılaşılmayacaktır.[1]

Bazı kaynaklarda ise Snoopy Bus protokolünden bahsedilmektedir. Bu protokol ile veri tutarlılığını sağlamak için Write Invalidate ve Write Update ilkeleri kullanılmaktadır. Write Invalidate ile ön bellekte işlenen, değiştirilen veri main memorideki yerine aktarılmakta Write Update ile de bu güncellenen veri diğer işlemcilerin ön belleklerine aktarılmakta böylece veri tutarsızlığı giderilmiş olmaktadır.[2]

Multicore işlmcilerde Cache Chorese protokolleri şunlardır: MSI protokolü, MESI protokolü, MOSI protokolü, MOESI protokolü, MERSI protokolü, MESIF protokolü, Write once protokolü, Synapse protokolü, Berkeley protokolü, Firefly protokolü, Dragon protokolü.[3]

MSI Protokolü

Cache Coherence de kullanılan en temel protokoldür. Bu protokole göre cach bellekte 3 tip blok vardır. Blokların tipi, depolamakta olduğu veri ve bu veriye yapılabilecek işlemler üzerinden sınıflandırılır.

Modified(M) : Bu blokta bulunan veriler üzerinde işlem yapılabilmektedir dolayısıyla veriler değişebilecek olan verilerdir ve bunların cache belleğe aktarılması çok önemlidir.

Shared(S) : Bu blokta bulunan veriler değiştirilmemiş olan verilerdir. Sadece okuma işlemi yapılmamış olan verilerdir. Cache bellekte bu veri main memorye göderilen kadar üzerinde hiçbir işlem yapılamaz.

Invalid(I) : Bu blok cache belekte geçersizdir. Eğer Cache de bulunuyorsa bu blok, başka bir cache bellek yada hafıza biriminden fetch edilmiştir. Anlaşılacağı üzere sadece veri aktarımı sağlanan bloktur.

Cache belleğe M yada S bloğu için okuma komutu geldiğinde, Eğer bu blok cachede I durumunda değil ise bu bloğun diğer cachelerdeki M bloklarında da bulunmadığı anlamına gelir. Farklı cache mimarilerinde bu durum farklı işlemektedir. Örneğin bus mimarisinde genellikle snooping kulanılmaktadır.

Diğer mimarilerde de cache directories bulunur ve burada da blokun, diğer cachelerde bulunan kopyalarının bilgisi tutulur. Eğer başka cachelerde de bu bloğun kopyası M durumunda mevcut ise o blok main memorye gönderilip yazılır, ardından I yada S durumuna getirilir. Eğer M bloğu mamin memorye geri yazılmış ise yada başka bir cachede S durumunda ise cache bellek bu bloğu alır. Ardından bu verinin okunması sağlanır ve bu bloğun durumu S state durumuna dönüşmüş olur.

Eğer, M bloğuna yazılmak üzere, bir yazma komutu cache belleğe gelirse cachede veri üzerine işlem yapılır. Fakat bu blok eğer S durumunda ise cache bu bloğun diğer cachelerdede S durumunda bulunup bulunmadığına bakar eğer bulunuyorsa S bloğu diğer cachelerden tahliye

edilir. Bu bus snooping yada directory üzerinden gerçekleştirilebiir. Ardından data üzerinde işlem yapılabilir ve blok M bloğuna dönüşmüş olur. Eğer blok I durumunda ise cache bellek diğer belleklerde bloğun S yada I durumunda olup olmadığına bakar. Eğer blok başka bir cachde M durumunda ise bloğun bulunduğu cache bloğu main memorye yada yazma komutunun gittiği cacheye aktarması gerekmektedir. Eğer blok diğer cachelerdee bulunmuyorsa ana hafızadan alınır ve üzerinde yazma işlemi gerçekleştirilir ardından blok M satate durumuna gelmiş olur.

Bu mimarinin uykulaması halinde farklı cacheler arasındaki blokların diğer bloklara erişimini aşşağıdaki tabloyu inceleyerek de görebilmekteyiz.[4]

	M	S	I
M	✗	✗	✓
S	✗	✓	✓
I	✓	✓	✓

MESI Protokolü

Write back i destekleyen en yaygın protokoldür. Illinois Üniversietsinde geliştirildiği için Illinois protokolü olarakda bilinmektedir. Write through metoduna göre harcanan hafıza kapasitesini kurtarabilmektedir.

Bu protokolde M,S ve I türündeki bloklara ek olarak *Exclusive*(E) türünde bir blok daha bulunmaktadır. E bloğundaki data sadece o cache bellekte bulunmaktadır. Aynı zamanda değitirilmemiş bir datadır. Yani bu datadaki veri Main memorydeki veri ile aynı veridir. Bu blok her an Modified yada Shared blok türüne dönüşebilir. Üzerine yazma komutu yada bloktan data okuma omutu gelecek olursa duruma göre M yada S durumuna dönüşecektir.

İki farklı cache bellek arasında bu blokların erişebilirliği akttaki tabloda görüleceği gibidir.

	M	E	S	I
M	✗	✗	✗	✓
E	✗	✗	✗	✓
S	✗	✗	✓	✓
I	✓	✓	✓	✓

Bir blok M yada E blok olduğu zaman diğer cache belleklerdede bu bloğun kopyası I bloğu olarak yerini almaktadır.

Blokların tür değişimi cache gelen 2 tür komuta, sinyale göre değişmektedir. Bunların ilki cach deki datalar üzerinde çalışan işlemciden cache belleğe gelen okuma yada yazma komutudur. Diğer komut, sinyal ise diğer işlemcilerden gelen komuttur. Diğer işlemcilerin bu data bloğuna sahip omadığı yada güncelleme yapılacağı zaman Bus iaralıcığı ile gönderilen komuttur. Yazma işlemi blok sadece M yada E durumunda ise direk gerçekleştirilebilir. Eğer blok S durumunda ise ilk olarak diğer tüm cachelerdeki bu bloğun kopyası geçersizleştirilir.[5]

MOSI Protokolü

Temelde MSI protokolünün bir uzantısıdır. Bu protokolde M, S ve I blok türlerine ek olarak Owned(O) bloğu bulunmaktadır. Mevcut işlemcinin bu bloğa sahip olduğunu, blok için diğer işlemcilerden gelen isteklere hizmet etiiğini gösterdiği söylenmektedir.

Birden fazla işlemci aynı bloğun, main memoride doğru olsada olmasada, doğru değerini tutuyoe olabilir. Aynı anda yalnızca bir cache bellek bir bloğun Owned durumuna sahip olabilir. Bu durumda diğer bütün önbelleklerde bu blok Shared durumundadır. M, S, I ve O durumlarına sahip olan farklı iki önbelleğin blokları arsındaki erişebilirliği aşşağıdaki tabloyuda inceleyerek anlayabilirirz[6]

	M	O	S	I
M	✗	✗	✗	✓
O	✗	✗	✓	✓
S	✗	✓	✓	✓
I	✓	✓	✓	✓

MOESI Protokolü

Bu protokol diğer protokollerde yaygın olarak kullanılan tüm olası blok durumlarını, blok türlerini içermektedir. 4 blok türüne sahip olan MESI protokolüne ek olarak, hem değiştirilen hem de paylaşılabilen blok olan O(owned) buloğunu içermektedir. Bu durum değiştirilen verilerin diğer cache bellekler ile paylaşılmadan önce main memorye yazılma

zorunluluğunu ortadan kaldırır. Veriler eninde sonunda main memorye yazılması gereksede , yazma ilemi ertelenebilir.

Tüm bu durumun gerçekleşebilmesi için verilerin, önbellekten ön belleğe doğrudan aktarılabilmesi gerekir. Böylece değiştirilmiş durumdaki verileri içeren bir cache bellek, bu verileri main memorye aktarmadan başka cache belleklere veriyi sağlamış olur.

M durumundaki blok, bloğun tek geçerli kopyasına sahipitir ve bu blokda değişiklikler yapılmıştır. O durumundaki blok diğer cache beleklerdede kopyası bulunablir fakat diğerlerinden bu blok sadece okunabilir, sadece bulunduğu cahche de işlem yapan işlemci değişiklik yapma hakkına sahiptir. Eğer üzerinde değişiklik yapılırsa diğer tüm cache belleklerede bildirilmesi ve güncellenmesi gerekmektedir. Böylece değiştirilmiş bir blok main memorye yazılmadan diğer cache bellekler arasında paylaşılmış olur. Yazma işlemi yapıldıktan sonra bu blok ya M durumuna geçer yada m durumndan maim memorye aktarılarak ardından S durumuna gelir. E bloğu cache belleğin bu bloğa sadece bir kopyası olarak sahip olduğunu gösterir. S bloğu de önceki protokolerde anlatıldığı gibi sadece okuma işlem yapılmış olan bloktur MESI protokolünden farklı olarak S bloğu değiştirilmiş olmaya bilir fakan başka bir cacahe O durumunda olarak değiştirilmiş olabilir. Bu protokole sahip olan mimaride iki farklı cache belleğin biribirlerinin bloklarına erişimini aşşağıdaki tabloyu inceleyerekde anlayabiliriz[7]

	M	O	E	S	I
M	✗	✗	✗	✗	✓
O	✗	✗	✗	✓	✓
E	✗	✗	✗	✗	✓
S	✗	✓	✗	✓	✓
I	✓	✓	✓	✓	✓

MERSI Protokolü

Bu protokol Power PC G4 tarafından kullanılan, Cache Coherence protokolüdür. Bu protokol 5 durumdan oluşur. Bu durumlar M, E, S, I ve Read only yada Recent (R) durumlarıdır. M, E, S, I durumları MESI protokolü ile aynıdır. R durumunda bulunan datanın kopyaları arasında tek değişirilmemiş data olması yönüyle E durumuna benzemektedir. E durumundan farklı olarak bloğun değiştirilip m durumuna geçebilmesi için R durumunda cache bellek

hattının sahipliğini istemesi gerekmektedir. Bu protokole sahip mimarideki iki işlemci arasında blok erişimini aşşağıdaki tabloyu inceleyerek de anlayabiliriz.[8]

	M	E	R	S	I
M	✗	✗	✗	✗	✓
E	✗	✗	✗	✗	✓
R	✗	✗	✗	✗	✓
S	✗	✗	✗	✓	✓
I	✓	✓	✓	✓	✓

MESIF Protokolü

Bu protokol Intel tarafında cacahe bellek uyumlu non uniform memory mimarileri için üretilmiştir. Protokolde 5 durumda olmak üzerek bloklar mevcuttur. M, E, S, I ve Forward(F). M, E, S, I durumları MESI protokoşü ile aynıdır. F bloğu S durumundaki blokların özelleştirilmiş bir biçimidir. Eğer Cacahe bellek bir bloğu S durumuna getirecek olursa başka bir blokta onun F türüne getirilmesi sağlanır.

MESI protokolüne sahip olan bir sistem birçok cache bellekde S durumda bloğun buunması sonucu verimsiz bir sonuç verilecektir. Main memoryden temin edilebilir yada bloğun paylaşıldığı bütün cache belleklere gereksiz olarak defalaca yanıt vererek yanıt bombardımanı da denilen duruma sebep olabilmektedir. MESIF protokolü kullanılan bir sistemde ise yalnızca hattı F durumunda tutan önbellek yanıt verecektir. Diğer belleklerinde yanıt vermesine gerek olmayacaktır. Bu durum önbellekler arasında daha hızlı bir akış olmasını sağlayacaktır. Cacahe belelk S yada F bloğunu tek taraflı olarak geçersiz kılabileceğinden, belleklerde S durumunda bloklar olabilecekken F durumunda hiç blok olamamsıda mümkündür. Bu durumda main memoryden istekte bulunarak daha verimli olması için son istekte bulunulan hatat F durumu verilecektir. Aşşağıdaki tabloyu inceleyerekde bu protokola sahip olan bir sistemde cache belelkler arasında blokların erişebiliriliğini gözlemleyebilirz.[9]

	M	E	S	I	F
M	✗	✗	✗	✓	✗
E	✗	✗	✗	✓	✗
S	✗	✗	✓	✓	✓
I	✓	✓	✓	✓	✓
F	✗	✗	✓	✓	✗

Write Once Protokolü

Cache Coherence protokolleri literatüründe tanımlanan ilk MESI protokolüdür. Data üzerinde ilk deişiklik, yazma işlemi gerçekleştiğinde datanın main memory ye de yazıldığı ardından gerçekleşen yazma datayı değiştirme işlemlerinde ise datanın meim memorye yazılmadığı, cache belelkten atıldığında en son son olarak main memorye yazıldığı optimizasyonuna sahiptpir. Burada amaç ardışık yazma işlemleri geldiğinde Bus da meydana gelecek olan veri yolu trafiği yoğunluğunu düşürmek ile birlikte sadece bir kere yazma işlemi yapılan verilerin güncel olarak main mermoride bulunmsını da sağlamaktır. İlk kez James R. Goodman tarafından 1983de tanımlanmıştır.

Bu protokolün kullanıldığı durumda 4 tip blok bulunabilmektedir. Invalid(I) bloğu belleğin turtarsız yani datanın diğer birimler ile yada main memory ile tutarsız olduğu durumun bir kopyasına sahiptir. Valid(V) bloğu bellekte tutarlı bölümün bir kopyasını içermektedir. Veriler muhtemelen paylaşılabilir ancak içeriği değiştirilemez. Reserved(R) bloğu ise belleğin tek kopyasıdır ve tutarlıdır. Eğer değiştirilecek olursa main memorye geri yazmaya gerek yoktur. Dirty(D) bloğu beleğin tek kopyasıdır ancak tutarsızdır. Bu blok bir yada daha fazla defa üzerinde işlem yapılarak yazılmıştır. Bu blok cache bellekte değiştirildiğinde write beck gerektiren tek durumdur.

Bu protokoldeki durumlar MESI protokolündeki durumlar ile aynı anlama gelmektedir. Ancak farklı olan ise bu protokolde tüm geçersiz kılma işlemleri main memorye geri yazma ile sonuçlanır. Bu protokolün uygulandığı sistemlerde cache bellekler arası blokların erişebilirliği aşşağıdaki tabloyu inceleyerek daha iyi anlaşılabilir.[10]

	I	V	R	D
I	✓	✓	✓	✓
V	✓	✓	✗	✗
R	✓	✗	✗	✗
D	✓	✗	✗	✗

Dragon Protokolü

Bu protokol, ön belleğe alınan tüm değerlerin doğrudan güncellenmesi ile gerçekleştirilir. Dragon protokolü gibi güncelleme tabanlı çalışan protokoller, bir blokta yazma işlemi gerçekleştirildikten sonra diğer işlemciler tarafından bu bloğa erişilmesi gerektiğinde, verimli bir şekilde çalışmaktadır .Çünkü blokda değişiklik yaptıktan sonra blok güncellenmiş ve bütün veriler güncel halde bulunmaktadır.

Bu protokolde 4 tür blok türü bulunmaktadır:

Exclusive clean(E) : Bu türdeki blok, bloğun mevcut bulunduğu cache üzerinde çalışan mevcut işlemci tarafından henüz ilkkez fetche edildiği, diğer işlemciler taradfından bu blok üzerinde erişim yapılmadığı anlamına gelmektedir.

Shared Clean(Sc) : Bu bloğun kesinlikle birden fazla işlemcinin cache belleğinde bulunduğu ve mevcut işlemcinin bu blok üzerinde yazma işlemini son yapan işlemci olmadığı anlamına gelmektedir. E ve Sc durumları, bus işlemleri tarafından, paylaşılmayan önbellek bloklarında okuma yazma işlemlerini önlemek için protokol tarafından ayrı ayrı tutulur. B udurumda çalışma hızını yavaşlatmaktadır. Bu tek threadli programlarda yaygın bir durumdur.

Shared Modified(Sm) : Bu bloğun birden çok işlemcinin cache belleğinde bulunduğu ve bulunduğu işlemcinin bu bloğu son değiştiren işlemci olduğu anlamına gelmektedir. Mevcut işlemciyi bloğun sahibi olarak adlandırabiliriz. Geçersiz kılma protokollerinin aksine bu bloğun Main Memoryde güncel olmasına gerek yoktur, yalnızca işlemcide güncel olması gerekir. Cache bellekteki bu blok tahliye edildiğinde Main Memoryde datayı güncellemei işlemcinin sorumluluğundadır.

Modify(M) : Yalnızca bir işlemcinin bu bloğa sahip olduğu, hafızadan getirildikten sonra üzerinde işlem yapıldığı anlamına gelmektedir.

Bu protokolün bulunduğu bir mimaride cacje bellekler arasında blokların erişebilirliğini aşşağıdaki tabloyu inceleyerek de daha iyi anlayabiliriz.[11]

	E	Sc	Sm	M
E	✗	✗	✗	✗
Sc	✗	✓	✓	✗
Sm	✗	✓	✗	✗
M	✗	✗	✗	✗

Firefly Protokolü

Bu protokol 3 durumlu Write Update Cache Coherence protokolüdür. Dragon protokolünün aksine Firefly protokolü, yazma işlemini Bus işlemleri ile Main Memoryi ve Cacahe belelkleri günceller. Bu protokolde blok türleri 3 tür olabilmektedir.

Valid Exclusive (V) : Bu blok geçerli, temiz yani değiştirilmemiş ve sadece bir cache bellekte bulunmaktadır.

Shared (S) : Bu blok geçerli, temiz yani değiştirilmemiştir. Bu blok birçok cache de bulunuyor olabilmektedir.

Dirty (D) : Bu blok hafızanın yalnızca bir kopyasıdır ve üzerinde işlem yapılmış olmakla birlikte değiştirilmiştir. Bu blok önbelelkde değiştirildiğinde geri yazma işlemi gereçekleştirilen yanı Main memorye yazılan tek bloktur.

Blok türlerinide inceledikten sonra bu mimarideki sistemde işleyiş mantığını basit olduğunu görmekteyiz. Sadece bir ache de bulunana ve işlenmemiş veri, Birçok bellekte bulunan işlenmemiş veri ve işlenmiş değiştirilmiş veri bulunduğunu değişim yapılan blokların güncellendiği ve önceside eğer S durumunda ise değişimden sonra D durumuna gelip ardından Main memoryde güncellenerek tekrara diğer önbelleklerde de güncellendiği eğer V durumnda ise ve diğer önbelleklerde bulunmuyorsa tekraradan diğer önbelleklere gidip güncelleme gereği olmadığı görerek mimarini çalışma mantığının basitliğini anlaya bilmekteyiz.[12]

KAYNAKÇA

[1] https://www.tutorialspoint.com/parallel_computer_architecture/parallel_computer_architecture_cache_coherence_synchronization.htm

[2] https://www.tutorialspoint.com/parallel_computer_architecture/parallel_computer_architecture_cache_coherence_synchronization.htm

[3] https://en.wikipedia.org/wiki/Cache_coherence

[4] https://en.wikipedia.org/wiki/MSI_protocol

[5] https://en.wikipedia.org/wiki/MESI_protocol

[6] https://en.wikipedia.org/wiki/MOSI_protocol

[7] https://en.wikipedia.org/wiki/MOESI_protocol

[8] https://en.wikipedia.org/wiki/MERSI_protocol

[9] https://en.wikipedia.org/wiki/MESIF_protocol

[10] https://en.wikipedia.org/wiki/Write-once_(cache_coherence)

[11] https://en.wikipedia.org/wiki/Dragon_protocol

[12] https://en.wikipedia.org/wiki/Firefly_(cache_coherence_protocol)

https://mgulercan.wordpress.com/2010/06/09/cache-coherence-in-turkish/