Haziran, 2026

BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ ALANINDA TEORİ VE UYGULAMALAR

EDİTÖR
DOÇ. DR. ARAFAT ŞENTÜRK

DOI: 10.5281/zenodo.21014900

Bilgisayar Bilimleri ve Mühendisliği Alanında Teori ve Uygulamalar

Editör

Doç. Dr. Arafat Şentürk

İmtiyaz Sahibi
Platanus Publishing®

Editör
Doç. Dr. Arafat Şentürk

Kapak & Mizanpaj & Sosyal Medya
Platanus Yayın Grubu

Birinci Basım
Haziran, 2026

Yayımcı Sertifika No
45813

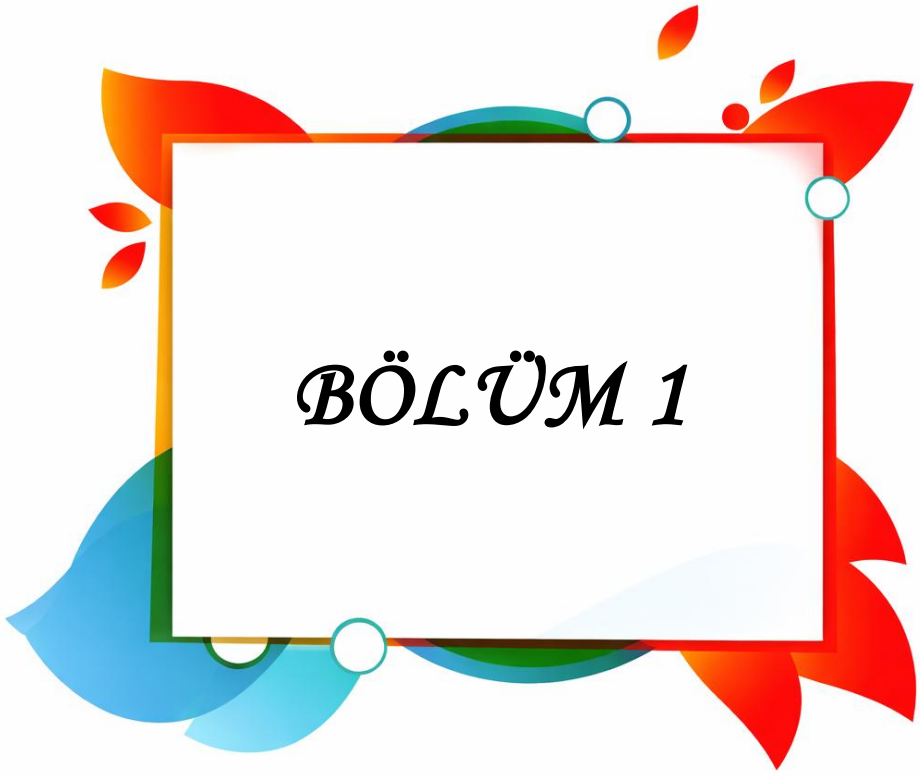
ISBN
978-625-6971-26-4

Adres: Natoyolu Cad. Fahri Korutürk Mah. 157/B, 06480, Mamak, Ankara, Türkiye.

Telefon: +90 312 390 1 118
web: www.platanuspublishing.com
e-mail: platanuskitap@gmail.com

Platanus Publishing®

İÇİNDEKİLER


BÖLÜM 1

Tam Homomorfik Şifreleme: Teorik Temeller, Modern Şemalar ve Uygulamalar

Nihat Eren Özmen[1] & Ercan Buluş[2]

1. Giriş

Dijital dönüşümün hızı, veriyi çok değerli haline getirirken, bu verinin güvenliği ve mahremiyeti modern bilişimin en büyük paradokslarından birini oluşturmuştur. Geleneksel kriptografi veriyi iletim ve depolama sırasında etkin bir şekilde korumaktadır. Ancak, bir algoritmanın veri üzerinde işlem yapabilmesi için verinin işlemci belleğinde açık hale getirilmesi zorunluluğu, "işlem anı" zafiyetini doğurmaktadır (Rivest, Adleman ve Dertouzos, 1978). Buna ek olarak, dijital ortamlarda verinin gizlenmesi ve korunmasına yönelik tekniklerin gelişmesi, bilgi güvenliği alanında yeni tehdit ve analiz ihtiyaçlarını da beraberinde getirmektedir; özellikle steganografi ve buna karşı geliştirilen steganaliz yaklaşımları, gizli bilgilerin tespiti açısından kritik bir araştırma alanı haline gelmiştir (Buluş, 2025).

Tam Homomorfik Şifreleme (Fully Homomorphic Encryption- FHE), bu paradoksu ortadan kaldıran bir teknolojidir. FHE, verinin deşifre edilmesine gerek kalmadan matematiksel fonksiyonların şifreli metinler üzerinde çalıştırılmasına olanak tanımaktadır. Bu işlem aşağıdaki (denklem 1) ifade ile elde edilmektedir:

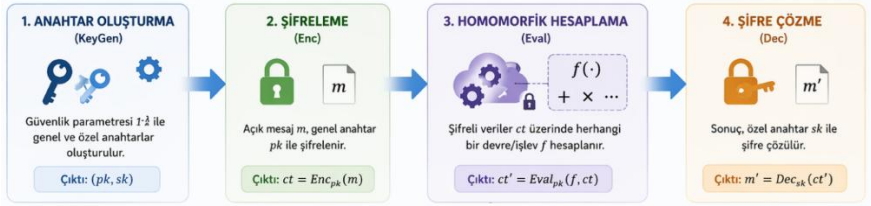
$$\mathrm{Dec}\left(\mathrm{f}\left(\mathrm{Enc}(\mathrm{m}_1), \mathrm{Enc}(\mathrm{m}_2), \ldots, \mathrm{Enc}(\mathrm{m}_n)\right)\right) = \mathrm{f}(\mathrm{m}_1, \mathrm{m}_2, \ldots, \mathrm{m}_n) \qquad (1)$$

Bu ifade, tam homomorfik şifreleme sistemlerinin doğruluk özelliğini tanımlamaktadır. Burada $\mathrm{m_i}$ değerleri açık veri uzayında tanımlı girdileri temsil ederken, Enc ve Dec sırasıyla şifreleme ve deşifreleme fonksiyonlarını göstermektedir. f fonksiyonu ise şifreli veri üzerinde homomorfik olarak uygulanabilen bir hesaplama operatörüdür.

Bu eşitlik, homomorfik değerlendirme işleminin doğruluğunu garanti etmekte olup, şifreli veri alanında gerçekleştirilen işlemlerin açık veri alanındaki karşılıkları ile tutarlı olduğunu ortaya koymaktadır. Bu özellik, özellikle gizlilik

[1] Arş. Gör., Tekirdağ Namık Kemal Üniversitesi, ORCID: 0000-0002-0053-3865
[2] Prof. Dr., Tekirdağ Namık Kemal Üniversitesi, ORCID: 0000-0001-9442-6253

koruyan veri işleme ve güvenli makine öğrenmesi uygulamaları açısından kritik öneme sahiptir. FHE sürecinin genel işleyişi Şekil 1'deki şematik gösterimde sunulmuştur.



Şekil 1. Tam Homomorfik Şifreleme yönteminin şematik gösterimi.

Son yıllarda FHE alanında yalnızca teorik gelişmeler değil, aynı zamanda performans ve uygulanabilirlik açısından önemli ilerlemeler kaydedilmiştir. Özellikle modern çalışmalar, FHE sistemlerinin donanımsal hızlandırma ve paralel hesaplama teknikleri ile gerçek dünya uygulamalarına daha yakın hâle geldiğini göstermektedir (Zhang vd., 2024; Fan vd., 2024).

2. Tarihsel Gelişim

Homomorfik şifreleme fikri ilk olarak Rivest, Adleman ve Dertouzos tarafından 1978 yılında ortaya atılmıştır (Rivest, Adleman ve Dertouzos, 1978). Bu çalışmada, şifreli veriler üzerinde belirli matematiksel işlemlerin gerçekleştirilebilmesi fikri teorik olarak tanımlanmış, ancak hem toplama hem de çarpma işlemlerini aynı anda ve sınırsız sayıda destekleyen bir sistemin gerçekleştirilmesi uzun yıllar boyunca mümkün olmamıştır. Bu nedenle erken dönem çalışmalar, yalnızca kısmi homomorfik özellikler sunan sistemler ile sınırlı kalmıştır.

Bu alandaki en önemli gelişme, ilk kez hem toplama hem de çarpma işlemlerini destekleyen ve teorik olarak sınırsız sayıda işlem yapılmasına olanak tanıyan bir tam homomorfik şifreleme sistemi önermiştir (Gentry, 2009). Bu yaklaşımın temel yeniliği, şifreli veriler üzerinde biriken gürültü problemini çözmek amacıyla geliştirilen bootstrapping (yeniden örnekleme) mekanizmasıdır. Bootstrapping, şifreli bir verinin deşifreleme işlemini yine şifreli ortamda çalıştırarak gürültü seviyesini yeniden başlangıç düzeyine indirmekte ve böylece sistemin sınırsız hesaplama yapabilmesini sağlamaktadır. Ancak bu işlem, yüksek hesaplama maliyeti nedeniyle pratik uygulamalarda önemli bir performans darboğazı oluşturmaktadır.

Gentry'nin çalışmasını takip eden dönemde, FHE sistemlerinin verimliliğini artırmaya yönelik araştırmalar gerçekleştirilmiş ve ikinci nesil olarak adlandırılan FHE yaklaşımları geliştirilmiştir. Bu kapsamda önerilen BGV (Brakerski–Gentry–Vaikuntanathan) ve BFV (Brakerski/Fan–Vercauteren) şemaları,

bootstrapping işlemini zorunlu kılmadan belirli derinlikte devrelerin hesaplanmasına olanak tanıyarak performans açısından önemli iyileştirmeler sunmuştur (Brakerski, Gentry ve Vaikuntanathan, 2014; Fan ve Vercauteren, 2012).

Daha sonraki çalışmalarda ise özellikle sayısal ve makine öğrenmesi uygulamalarına yönelik ihtiyaçlar doğrultusunda CKKS (Cheon–Kim–Kim–Song) şeması geliştirilmiştir. CKKS, yaklaşık aritmetik işlemleri desteklemesi sayesinde gerçek sayılar üzerinde verimli hesaplama yapılmasına olanak tanımakta ve bu özelliği ile gizlilik koruyan yapay zekâ uygulamalarında yaygın olarak kullanılmaktadır (Cheon, Kim, Kim ve Song, 2017). Buna ek olarak, TFHE (Fast Fully Homomorphic Encryption over the Torus- Torus üzerinde Hızlı Tam Homomorfik Şifreleme) şeması, bit düzeyinde işlemler ve hızlı bootstrapping mekanizması ile düşük gecikmeli uygulamalar için önemli avantajlar sunmaktadır (Chillotti, Gama, Georgieva ve Izabachene, 2016). Güncel çalışmalar, CKKS şemasının özellikle GPU tabanlı hızlandırma ile önemli performans kazanımları elde edebildiğini göstermektedir. Bu bağlamda, CKKS tabanlı sistemlerin yüksek hacimli veri işleme ve makine öğrenmesi uygulamalarında etkin bir şekilde kullanılabildiği ortaya konulmuştur (Özcan ve Savaş, 2024; Fan vd., 2024).

3. Matematiksel Temeller ve LWE Problemi

Modern tam homomorfik şifreleme şemalarının büyük bir bölümü, kafes tabanlı kriptografi ve özellikle Hatalarla Öğrenme (Learning With Errors- LWE) problemine dayanmaktadır. LWE problemi hem klasik hem de kuantum bilgisayarlara karşı dirençli olması nedeniyle günümüzde kuantum sonrası kriptografinin temel yapı taşlarından biri olarak kabul edilmektedir (Regev, 2009).

Kafes tabanlı kriptografi, güvenliğini yüksek boyutlu matematiksel kafes yapıları üzerinde tanımlanan zor problemlere dayandıran bir kriptografik yaklaşımdır. Bu yapılar, çok boyutlu uzayda belirli doğrusal kombinasyonlarla elde edilen nokta kümeleri olarak tanımlanmakta olup, bu uzayda en kısa vektörün bulunması veya belirli bir noktaya en yakın kafes noktasının hesaplanması gibi problemler hesaplama açısından son derece zordur. Bu zorluk, klasik ve kuantum algoritmalarına karşı dayanıklılık sağladığı için kafes tabanlı yöntemler kuantum sonrası kriptografinin en güçlü adayları arasında yer almaktadır. Ayrıca bu yapı, gürültü temelli modeller ile birleştirildiğinde, özellikle LWE tabanlı şemalarda hem güvenlik hem de uygulanabilirlik açısından önemli avantajlar sunmaktadır.

LWE problemi ise bu zor problemlerin doğrudan kullanılmasından ziyade, daha esnek ve uygulanabilir bir yapı sunmaktadır. Temel olarak LWE, doğrusal denklem sistemlerine kontrollü bir rastgele gürültü eklenmesi ile oluşturulmaktadır. Bu sayede, dışarıdan gözlemlenen veriler istatistiksel olarak rastgele görünmekte ve gizli anahtarın elde edilmesi önemli ölçüde zorlaşmaktadır. Ayrıca bu gürültü mekanizması hem güvenliği artırmakta hem de homomorfik işlemler sırasında oluşan hata birikiminin matematiksel olarak yönetilebilmesine olanak tanımaktadır. Bu yönüyle LWE, teorik olarak zor kabul edilen kafes problemleri ile pratik kriptografik uygulamalar arasında bir köprü oluşturmakta ve modern FHE şemalarının temelini oluşturmaktadır.

3.1. LWE ve Gürültü Dinamiği

LWE problemi, gürültü içeren doğrusal denklemlerden gizli bir vektörün elde edilmesini konu almaktadır ve aşağıdaki (denklem 2) şekilde ifade edilir:

$$b_i = \langle a_i, s \rangle + e_i (\mathrm{mod} q) \qquad (2)$$

Burada a_i, n boyutlu rastgele seçilmiş bir vektörü, s gizli anahtar vektörünü ve e_i küçük büyüklükte, genellikle belirli bir olasılık dağılımından (örneğin Gauss dağılımı) örneklenen gürültü terimini temsil etmektedir. q ise genellikle büyük bir asal sayı veya uygun bir modül değeridir.

Bu yapı, klasik doğrusal denklem sistemlerinden temel bir noktada ayrılmaktadır. Gürültü terimi e_i'nin varlığı, sistemi doğrudan çözülmesi mümkün olan deterministik bir yapıdan çıkararak, istatistiksel olarak belirsiz ve hesaplama açısından zor bir probleme dönüştürmektedir. Gürültü teriminin sıfır olması durumunda problem basit bir lineer cebir problemine indirgenebilirken, gürültünün varlığı bu çözümü pratikte imkânsız hâle getirmektedir.

Ayrıca bu gürültü bileşeni yalnızca güvenliği sağlamakla kalmayıp, aynı zamanda homomorfik şifreleme şemalarında hesaplama sırasında biriken hata terimlerinin modellenmesine de olanak tanımaktadır. Ancak bu durum, gürültünün kontrol edilmesini kritik hâle getirmekte ve gürültü büyümesi (noise growth) problemini ortaya çıkarmaktadır. Bu nedenle modern FHE şemalarında, gürültünün sınırlandırılması ve gerektiğinde azaltılması için çeşitli teknikler (örneğin yeniden ölçekleme ve bootstrapping işlemleri) kullanılmaktadır.

3.2. LWE Tabanlı Şifreleme Mekanizması

LWE problemi, doğrudan bir şifreleme mekanizmasına dönüştürülebilmektedir. Bu bağlamda bir mesajın şifrelenmesi (denklem 3) ile ifade edilir:

$$\mathrm{Enc}(m) = (a, b = \langle a, s \rangle + e + m \ (\mathrm{mod}\ q)) \qquad (3)$$

Burada a rasgele seçilmiş bir vektörü, b şifreli değeri, s gizli anahtar vektörünü, e küçük bir rasgele gürültü terimini ve m ise genellikle küçük bir değerde olan mesajı temsil edilmekte ve gürültü terimi ile şifreli veri içerisine gömülmektedir.

Deşifreleme işlemi ise aşağıdaki (denklem 4) şekilde gerçekleştirilir:

$$m' = b - \langle a, s \rangle \pmod{q} \quad (4)$$

Bu işlem sonucunda elde edilen değer (denklem 5):

$$m' = m + e \quad (5)$$

şeklinde olup, e yeterince küçük olduğu sürece doğru mesaj elde edilmektedir. Bu durum, sistem doğruluğunun doğrudan gürültü kontrolüne bağlı olduğunu göstermektedir.

3.3. Gürültü Dinamiği ve Homomorfik İşlemler

FHE sistemlerinde şifreli veriler üzerinde gerçekleştirilen homomorfik işlemler sırasında, şifreli metinlere gömülü olan gürültü terimi kademeli olarak artmaktadır. Gürültü artışı, gerçekleştirilen işlemin türüne bağlı olarak farklı davranışlar sergilemektedir.

Toplama işlemi

İki şifreli veri toplandığında, toplam gürültü yaklaşık olarak doğrusal bir şekilde artmaktadır (denklem 6):

$$e_{\text{toplam}} \approx e_1 + e_2 \quad (6)$$

Bu nedenle toplama işlemleri, gürültü büyümesi açısından görece daha kontrollü olup, sistemin doğruluğunu önemli ölçüde etkilememektedir.

Çarpma işlemi

Homomorfik çarpma işlemi ise gürültü büyümesi açısından kritik bir rol oynamaktadır. Çarpma işlemi sonucunda oluşan gürültü, yalnızca mevcut gürültü terimlerinin etkileşimiyle değil, aynı zamanda mesaj ve anahtar bileşenleri ile olan çapraz terimlerden dolayı daha hızlı artmaktadır. Bu durum yaklaşık olarak (denklem 7):

$$e_{\text{çarpım}} \gg e_1 + e_2 \quad (7)$$

şeklinde ifade edilebilir.

Dolayısıyla çarpma işlemleri, gürültünün doğrusal değil, daha hızlı (çoğu durumda katlanarak) büyümesine neden olmaktadır.

Genel gürültü davranışı

Birden fazla homomorfik işlem gerçekleştirildiğinde, toplam gürültü birikimli olarak artmakta ve hesaplama derinliği ile doğrudan ilişkili hâle gelmektedir. Bu nedenle, belirli bir eşik değerin aşılması durumunda deşifreleme hataları ortaya çıkabilmektedir.

Bu bağlamda FHE sistemlerinde:

- gürültü büyümesinin kontrol edilmesi
- hesaplama derinliğinin sınırlandırılması
- gerektiğinde bootstrapping gibi tekniklerin kullanılması

kritik öneme sahiptir.

3.4. Gürültü Eşiği ve Doğruluk Koşulu

Şifreli verinin doğru şekilde çözülebilmesi için, şifreli metin içerisinde bulunan toplam gürültü teriminin belirli bir eşik değerinin altında kalması gerekmektedir. Bu koşul genel olarak aşağıdaki şekilde (denklem 8) ifade edilmektedir:

$$|e| < \frac{q}{2} \tag{8}$$

Bu eşitsizlikte e gürültüyü, q modülü temsil etmektedir. Deşifreleme sırasında modüler aritmetik kapsamında oluşabilecek taşmaların önlenmesi açısından kritik bir sınırı temsil etmektedir. Gürültü teriminin bu sınırı aşması durumunda, mesaj bileşeni ile gürültü birbirine karışmakta ve doğru mesajın geri elde edilmesi mümkün olmamaktadır.

Pratikte, mesajlar genellikle belirli bir ölçekleme faktörü ile modül uzayına yerleştirildiğinden, doğruluk koşulu yalnızca gürültü büyüklüğüne değil, aynı zamanda mesaj temsiline de bağlıdır. Bu nedenle FHE sistemlerinde:

- modül değeri q
- gürültü dağılımının parametreleri
- gerçekleştirilecek işlem derinliği

birlikte dikkate alınarak uygun parametre seçimi yapılmalıdır.

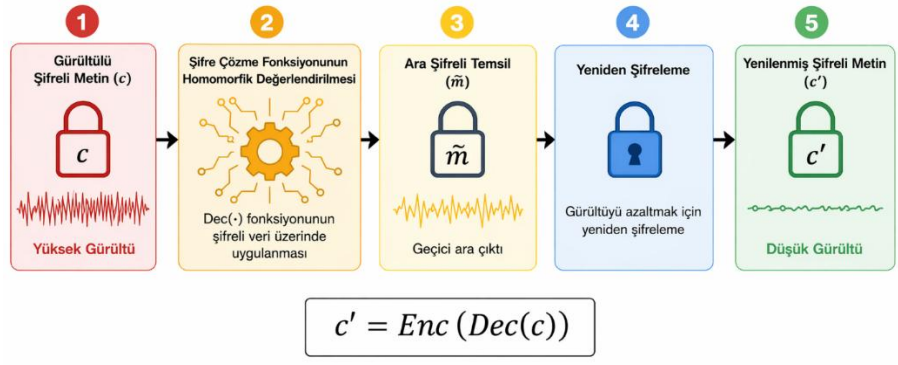
3.5. Bootstrapping Mekanizması

Bootstrapping, FHE sistemlerinde biriken gürültü seviyesini kontrol altına almak ve hesaplama derinliğini artırmak amacıyla kullanılan temel bir tekniktir. Kavramsal olarak bu işlem aşağıdaki şekilde (denklem 9) ifade edilebilir:

$$c' = \mathrm{Enc}(\mathrm{Dec}(c)) \tag{9}$$

Bu ifade, bootstrapping işleminin temel mantığını göstermektedir. Buna göre, şifreli veri c üzerinde tanımlı olan deşifreleme fonksiyonu, yine şifreli alan içerisinde homomorfik olarak uygulanmakta ve elde edilen sonuç yeniden şifrelenmektedir. Bu süreçte veri hiçbir aşamada açık hâle getirilmeden işlem görmektedir.

Başka bir ifadeyle, sistem kendi deşifreleme algoritmasını şifreli veri üzerinde çalıştırarak mevcut gürültü bileşenini azaltmakta ve yeni bir şifreli veri c' üretmektedir. Elde edilen bu yeni şifreli veri, daha düşük gürültü seviyesine sahip olduğu için ek homomorfik işlemlerin gerçekleştirilmesine olanak tanımaktadır. Bu sayede gürültü bütçesi (noise budget) yeniden kazanılmakta ve sistemin işlem kapasitesi artırılmaktadır. Bootstrapping sürecinin işleyişi Şekil 2'de sunulmuştur.



Şekil 2. Bootstrapping mekanizmasının şematik gösterimi.

Bu mekanizma sayesinde, gürültü birikimi nedeniyle sınırlanan hesaplama derinliği artırılmakta ve FHE sistemleri teorik olarak sınırsız sayıda işlem gerçekleştirebilecek hâle getirilmektedir. Bu yönüyle bootstrapping, tam homomorfik şifreleme sistemlerinin pratik uygulanabilirliğini belirleyen en kritik bileşenlerden biri olarak değerlendirilmektedir.

Ancak bootstrapping işlemi hesaplama açısından oldukça maliyetlidir ve yüksek işlem süresi gerektirmektedir. Bu nedenle modern FHE sistemlerinde, bootstrapping ihtiyacını azaltmak amacıyla:

- leveled FHE yaklaşımları
- batching (SIMD) teknikleri
- parametre optimizasyonu

gibi yöntemler geliştirilmiştir.

4. Modern FHE Şemalarının Karşılaştırılması

Günümüzde tam homomorfik şifreleme alanında yaygın olarak kullanılan başlıca şemalar BGV, BFV, CKKS ve TFHE'dir. Bu şemalar, destekledikleri aritmetik türü, gürültü yönetimi yaklaşımları ve uygulama alanları açısından birbirlerinden ayrılmaktadır.

BGV şeması, modüler tam sayılar üzerinde çalışan ve gürültü yönetimini modül küçültme (modulus switching) teknikleri ile gerçekleştiren bir yapıya sahiptir. Bu şema, teorik olarak güçlü güvenlik temellerine dayanmakta olup, özellikle derin devrelerin belirli bir seviyeye kadar hesaplanmasında etkili sonuçlar sunmaktadır.

BFV şeması ise tam sayılar üzerinde kesin (exact) aritmetik işlemleri desteklemektedir. Bu nedenle finansal hesaplamalar ve doğruluk gerektiren uygulamalar için daha uygun bir yapı sunmaktadır. BFV'de mesajlar doğrudan tamsayı olarak temsil edilmekte ve gürültü yönetimi dikkatli parametre seçimi ile sağlanmaktadır.

CKKS şeması, diğerlerinden farklı olarak yaklaşık aritmetik (approximate arithmetic) üzerine kuruludur. Bu şema, gerçek ve kompleks sayılar üzerinde hesaplama yapılmasına olanak tanımakta ve özellikle kayan noktalı işlemlerin gerekli olduğu makine öğrenmesi ve veri analitiği uygulamalarında önemli avantajlar sağlamaktadır. CKKS'te hesaplamalar sırasında belirli bir hata toleransı kabul edilmekte olup, bu durum performans ve verimlilik açısından önemli bir kazanım sunmaktadır. Bu nedenle CKKS, gizlilik koruyan makine öğrenmesi uygulamalarında en yaygın kullanılan FHE şemalarından biri olarak öne çıkmaktadır.

TFHE şeması ise bit düzeyinde işlemler üzerine odaklanmakta ve özellikle hızlı bootstrapping mekanizması ile dikkat çekmektedir. Bu yapı, mantıksal kapılar (AND, OR, NOT) üzerinden işlem yapılmasını mümkün kılmakta ve düşük gecikmeli uygulamalarda avantaj sağlamaktadır. TFHE, özellikle gerçek zamanlı ve düşük gecikme gerektiren senaryolarda tercih edilmektedir.

Bu şemalar arasındaki temel farklar; desteklenen veri tipi (tam sayı, yaklaşık sayı, bit düzeyi), hesaplama doğruluğu, gürültü yönetimi ve performans özellikleri üzerinden değerlendirilmektedir. Uygulama alanına bağlı olarak uygun şemanın seçilmesi, sistem performansı ve doğruluğu açısından kritik öneme sahiptir. Tablo 1'de modern tam homomorfik şifreleme şemalarının veri tipi, aritmetik işlem türü, gürültü yönetim tekniğine ve kullanım alanlarına göre karşılaştırılması yer almaktadır.

Tablo 1. Modern tam homomorfik şifreleme şemalarının karşılaştırılması

Şema	Veri Tipi	Aritmetik Türü	Gürültü Yönetim Tekniği	Kullanım Alanı
BGV	Tam sayılar	Kesin (Exact)	Modül küçültme (Modulus Switching)	Genel kriptografik uygulamalar
BFV	Tam sayılar	Kesin (Exact)	Modül küçültme + Yeniden doğrusal hâle getirme (Relinearization)	Finansal hesaplamalar
CKKS	Reel/Karmaşık sayılar	Yaklaşık (Approximate)	Yeniden ölçekleme (Rescaling)	Makine öğrenmesi ve veri analitiği
TFHE	Bit düzeyi	Mantıksal işlemler	Hızlı önyükleme (Fast Bootstrapping)	Düşük gecikmeli uygulamalar

5. Tam Homomorfik Şifreleme ile Mahremiyeti Koruyan Yapay Zekâ Uygulamaları

Tam homomorfik şifreleme, verilerin şifreli haldeyken işlenebilmesine olanak tanıyarak, mahremiyeti koruyan yapay zekâ (Privacy-Preserving Machine Learning - PPML) alanında önemli bir çözüm sunmaktadır. Geleneksel makine öğrenmesi yaklaşımlarında, model eğitimi veya çıkarım aşamalarında verilerin açık hâlde işlenmesi gerekmekte olup, bu durum özellikle hassas veriler açısından ciddi güvenlik ve gizlilik riskleri oluşturmaktadır. FHE sayesinde ise veriler hiçbir aşamada deşifre edilmeden işlenebilmekte ve böylece veri gizliliği korunmaktadır.

5.1. Şifreli Veri Üzerinde Makine Öğrenmesi

FHE tabanlı sistemlerde en yaygın kullanım senaryolarından biri, şifreli veri üzerinde çıkarım yapılmasıdır. Bu yaklaşımda kullanıcı, verisini şifreleyerek bir sunucuya gönderir ve sunucu bu veri üzerinde eğitilmiş modeli kullanarak tahmin işlemini gerçekleştirir. Elde edilen sonuç yine şifreli olarak kullanıcıya iletilir ve yalnızca kullanıcı tarafından deşifre edilir.

Bu yapı sayesinde:

- veri hiçbir zaman açık hâle getirilmez
- sunucu veri içeriğine erişemez
- model ile veri arasında güvenli bir etkileşim sağlanır.

Bu yaklaşım özellikle bulut tabanlı yapay zekâ hizmetlerinde kritik bir öneme sahiptir.

5.2. CKKS Şeması ve Yaklaşık Hesaplama Avantajı

Makine öğrenmesi uygulamalarında çoğunlukla reel sayılar ve kayan noktalı işlemler kullanılmaktadır. Bu nedenle CKKS şeması, yaklaşık aritmetik yapısı sayesinde PPML uygulamalarında öne çıkmaktadır. CKKS, belirli bir hata toleransı ile hesaplama yapılmasına izin vererek performans açısından önemli kazanımlar sağlamaktadır.

Bu özellikler sayesinde:

- lineer cebir işlemleri (matris çarpımı, vektör işlemleri)
- sinir ağı hesaplamaları
- regresyon ve sınıflandırma algoritmaları

şifreli veri üzerinde uygulanabilmektedir. Yaklaşık hesaplama yaklaşımı, özellikle derin öğrenme modellerinde yüksek verimlilik sağlamaktadır.

5.3. Uygulama Alanları

FHE tabanlı mahremiyeti koruyan yapay zekâ uygulamaları, özellikle hassas veri içeren alanlarda büyük önem taşımaktadır:

- Sağlık: Hasta verileri üzerinde gizlilik ihlali olmadan teşhis ve analiz yapılabilmesi
- Finans: Müşteri verileri üzerinde güvenli risk analizi ve kredi skorlama
- Bulut bilişim: Kullanıcı verilerinin servis sağlayıcıya açılmadan işlenmesi
- Savunma ve güvenlik: Kritik verilerin korunarak analiz edilmesi

Bu alanlarda FHE, veri paylaşımı ile gizlilik arasında bir denge kurulmasını sağlamaktadır.

5.4. Karşılaşılan Zorluklar ve Sınırlamalar

FHE tabanlı PPML sistemleri önemli avantajlar sunmakla birlikte, bazı temel zorluklar da içermektedir. FHE sistemlerinde performansın en büyük darboğazlarından biri, bellek erişimi ve yüksek dereceli polinom işlemleridir. Yapılan güncel çalışmalar, özellikle CKKS şemasında performansın büyük ölçüde bellek bant genişliği ve paralel işlem kapasitesi ile sınırlı olduğunu göstermektedir (Choi vd., 2025). Temel zorluklar aşağıda yer alan şekilde sıralanabilir:

- Yüksek hesaplama maliyeti: Şifreli işlemler klasik hesaplamalara kıyasla önemli ölçüde daha yavaştır. Örneğin, açık veri üzerinde milisaniye seviyesinde gerçekleştirilebilen bir çarpma işlemi, FHE ortamında saniyeler mertebesine

kadar uzayabilmektedir. Bu durum, özellikle derin öğrenme modellerinde performans açısından ciddi sınırlamalar oluşturmaktadır.

- Gürültü birikimi: Derin hesaplama süreçlerinde gürültü artışı performansı sınırlar.
- Model kısıtları: Aktivasyon fonksiyonları gibi bazı işlemler doğrudan uygulanamaz.
- Bellek ve bant genişliği gereksinimi: Şifreli veriler büyük boyutlara ulaşabilmektedir.

Bu sınırlamaları aşmak amacıyla, son yıllarda hem yazılımsal hem de donanımsal düzeyde önemli gelişmeler kaydedilmiştir.

- Intel tarafından geliştirilen HEXL kütüphanesi (Homomorphic Encryption Acceleration Library), düşük seviyeli aritmetik işlemleri optimize ederek FHE hesaplamalarının hızlandırılmasına katkı sağlamaktadır (Boemer vd., 2021).
- Microsoft tarafından geliştirilen SEAL kütüphanesi, BFV ve CKKS şemaları için optimize edilmiş uygulamalar sunarak araştırmacılar ve geliştiriciler için güçlü bir altyapı sağlamaktadır (Microsoft SEAL, 2023).
- GPU ve FPGA tabanlı hızlandırıcılar ile, FHE işlemlerinin paralel olarak yürütülmesi mümkün hâle gelmiştir.
- Ayrıca son dönemde, FHE işlemlerine özel ASIC (Application-Specific Integrated Circuit) tasarımları üzerinde çalışmalar yürütülmekte olup, bu yaklaşımlar FHE'nin pratik uygulanabilirliğini artırma potansiyeline sahiptir.

Bu gelişmeler, FHE sistemlerinin performans sınırlamalarını azaltmaya yönelik önemli adımlar olarak değerlendirilmekte ve gelecekte daha geniş ölçekli uygulamaların önünü açmaktadır. Donanımsal hızlandırma alanında yapılan son çalışmalar, özel amaçlı FHE hızlandırıcılarının (ASIC) genel amaçlı CPU ve GPU mimarilerine kıyasla çok daha yüksek performans sunduğunu ortaya koymaktadır. Bu tür mimariler, özellikle sayı-teorik dönüşümler (NTT) ve modüler aritmetik işlemleri için optimize edilerek önemli hız kazançları sağlamaktadır (Zhang vd., 2024).

6. Sonuç ve Gelecek Çalışmalar

Bu çalışmada, tam homomorfik şifreleme sistemlerinin matematiksel temelleri, gürültü dinamiği ve modern şemaları kapsamlı bir şekilde incelenmiştir. Özellikle kafes tabanlı kriptografi ve LWE problemi üzerinden geliştirilen FHE yapılarının, şifreli veri üzerinde doğrudan hesaplama

yapılabilmesine olanak tanıdığı gösterilmiştir. Bu bağlamda, gürültü büyümesi, doğruluk koşulu ve bootstrapping mekanizması gibi temel kavramlar detaylı olarak ele alınmıştır.

Çalışma kapsamında ayrıca BGV, BFV, CKKS ve TFHE gibi modern FHE şemaları karşılaştırılmış ve bu şemaların veri tipi, aritmetik yapısı ve uygulama alanları açısından farklılıkları ortaya konulmuştur. Özellikle CKKS şemasının yaklaşık aritmetik yapısı sayesinde makine öğrenmesi uygulamalarında önemli bir avantaj sağladığı vurgulanmıştır.

FHE'nin mahremiyeti koruyan yapay zekâ (PPML) uygulamalarındaki rolü incelendiğinde, şifreli veri üzerinde güvenli çıkarım yapılabilmesinin veri gizliliği açısından kritik bir çözüm sunduğu görülmektedir. Sağlık, finans ve bulut bilişim gibi alanlarda, verilerin açık hâle getirilmeden işlenebilmesi, modern veri güvenliği yaklaşımları açısından önemli bir paradigma değişimini temsil etmektedir.

Bununla birlikte, FHE sistemlerinin yaygın kullanımının önünde hâlen bazı önemli zorluklar bulunmaktadır. Yüksek hesaplama maliyeti, gürültü birikimi, sınırlı işlem derinliği ve büyük veri boyutları, bu teknolojinin pratik uygulamalarda kullanımını kısıtlayan başlıca faktörlerdir. Bu nedenle performans optimizasyonu ve gürültü yönetimi, FHE araştırmalarının odak noktalarından biri olmaya devam etmektedir.

Gelecek çalışmalar açısından, daha verimli bootstrapping algoritmalarının geliştirilmesi, donanımsal hızlandırma (GPU/FPGA) çözümlerinin entegrasyonu ve hibrit kriptografik yaklaşımların kullanımı önemli araştırma yönleri olarak öne çıkmaktadır. Ayrıca FHE ile diferansiyel gizlilik, güvenli çok taraflı hesaplama ve federated learning gibi yöntemlerin birlikte kullanılması, daha güçlü ve esnek gizlilik koruma mekanizmalarının geliştirilmesine katkı sağlayacaktır.

Sonuç olarak, tam homomorfik şifreleme, veri gizliliğini koruyarak hesaplama yapılmasını mümkün kılan devrimsel bir teknoloji olarak öne çıkmakta ve özellikle yapay zekâ uygulamalarında geleceğin güvenli veri işleme paradigmalarından biri olarak değerlendirilmektedir.

KAYNAKÇA

Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2014). (Leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT), 6(3), 1-36.

Bulus, E. (2024). Investigation of steganalysis performance of selected steganography methods with deep learning models. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *40*(1).

Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017, November). Homomorphic encryption for arithmetic of approximate numbers. In International conference on the theory and application of cryptology and information security (pp. 409-437). Cham: Springer International Publishing.

Chillotti, I., Gama, N., Georgieva, M., & Izabachene, M. (2016, November). Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In international conference on the theory and application of cryptology and information security (pp. 3-33). Berlin, Heidelberg: Springer Berlin Heidelberg.

Choi, W., Yu, H., Kim, J., Ji, H., Park, J., & Ahn, J. H. (2025). Theodosian: A Deep Dive into Memory-Hierarchy-Centric FHE Acceleration. arXiv preprint arXiv:2512.18345.

Fan, J., & Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive.

Fan, S., Deng, X., Tian, Z., Hu, Z., Chang, L., Hou, R., ... & Zhang, M. (2024). Taiyi: A high-performance CKKS accelerator for practical fully homomorphic encryption. arXiv preprint arXiv:2403.10188.

Gentry, C. (2009, May). Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing (pp. 169-178).

Boemer, F., Kim, S., Seifu, G., DM de Souza, F., & Gopal, V. (2021, November). Intel HEXL: accelerating homomorphic encryption with Intel AVX512-IFMA52. In Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography (pp. 57-62).

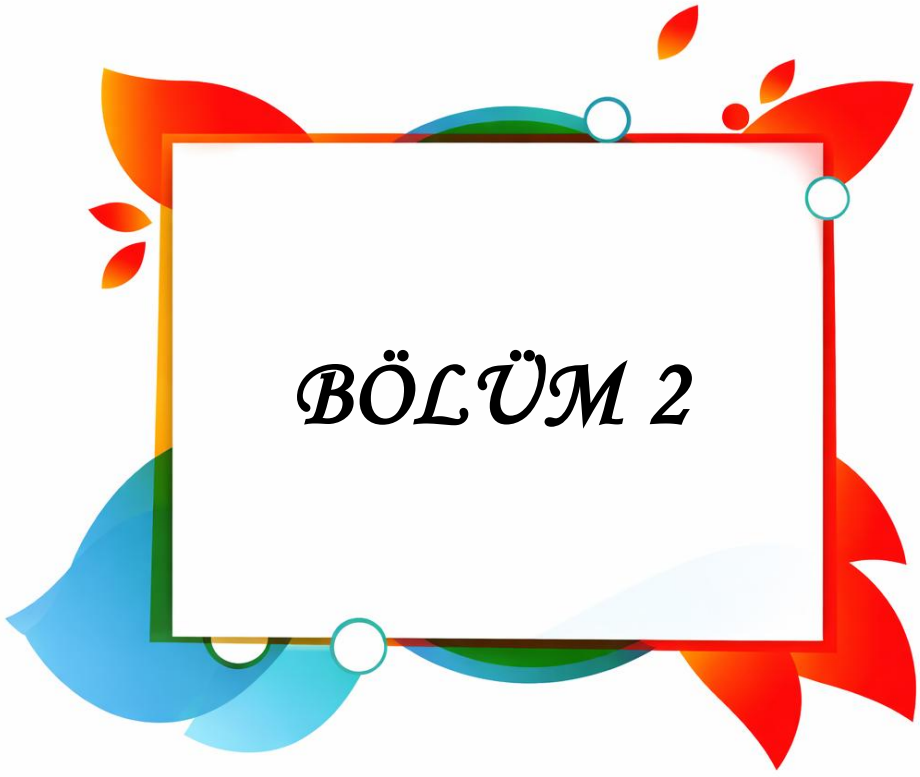
Microsoft. (2023). Microsoft SEAL (Version 4.1) [Computer software].

Özcan, A. Ş., & Savaş, E. (2024). HEonGPU: a GPU-based fully homomorphic encryption library 1.0. Cryptology ePrint Archive.

Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM), 56(6), 1-40.

Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. Foundations of secure computation, 4(11), 169-180.

Zhang, J., Cheng, X., Yang, L., Hu, J., Liu, X., & Chen, K. (2024). Sok: Fully homomorphic encryption accelerators. ACM Computing Surveys, 56(12), 1-32.


BÖLÜM 2

Yüksek Yoğunluklu Çok Çekirdekli Sistemlerde Sanal Bellek Optimizasyonu

Ruhi Taş[1]

I. GİRİŞ

Tek çekirdekli işlemcilerden çip başına onlarca veya yüzlerce çekirdeğe sahip sistemlere geçiş, bellek sistemleri üzerindeki baskıyı köklü biçimde dönüştürmüştür. İşlemciler hızlanmaya devam ederken bellek bant genişliği ve gecikmesi bu artışa yetişememiş; "bellek duvarı" olarak adlandırılan bu asimetri, işlemcinin verileri beklemek zorunda kaldığı kronik bir darboğaza dönüşmüştür. Bu nedenle, genellikle görünmez kalan sanal bellek alt sistemi, modern sunuculardaki kritik performans kısıtlayıcısı konumuna yükselmiştir [1, 2].

Sanal bellek, modern işletim sistemleri için vazgeçilmezdir: her programa fiziksel bellekten bağımsız büyük ve sürekli bir adres uzayı sunar, işlemler arasında izolasyon sağlar, kod paylaşımına olanak tanır ve isteğe bağlı sayfalama (demand paging) gibi tekniklerin kullanılmasına zemin hazırlar [3]. Ancak bu esneklik bedelsiz değildir; her bellek erişiminde sanal adresin fiziksel adrese çevrilmesi gerekmektedir. Yüksek yoğunluklu çok çekirdekli sistemlerde bu adres dönüşümünün, TLB yönetiminin ve sayfa hatası işlemenin getirdiği maliyetler ciddi bir performans kaybına yol açabilmektedir [4].

Çekirdek sayısı arttıkça çok çekirdeğe özgü birkaç problem bu maliyetleri daha da ağırlaştırmaktadır:

- **TLB Shootdown Yükü:** İşletim sistemi bir eşlemeyi değiştirdiğinde (sayfa kaldırma veya izin güncelleme), tüm çekirdeklerdeki eski TLB girişlerini geçersiz kılmak için Çekirdeklerarası Kesmeler (IPI) gönderilmelidir. Onlarca veya yüzlerce çekirdeğe sahip sistemlerde bu shootdown'lar gecikmede ani artışlara ve önemli çekirdek yüküne neden olmaktadır.
- **Sayfa Tablosu Çekişmesi:** Aynı işlemi paylaşan birden fazla iş parçacığı eşzamanlı olarak sayfa hatalarına neden olabilir ya da eşlemeleri güncellemeye çalışabilir. Bu yapıları korumak için kullanılan kilitler, çekirdek sayısı ile birlikte aynı kilitleri bekleyen iş

[1] Dr., Ostim Teknik Üniversitesi, Yazılım Mühendisliği Bölümü

parçacıklarının oluşturduğu serileştirmeyi ve önbellek hattı sıçramasını artırmaktadır.

- **Toplam Çalışma Kümesi Büyümesi:** Çok sayıda programın birlikte çalışmasıyla ortaya çıkan toplam bellek gereksinimi, TLB ve paylaşılan önbelleklerin kapasitesini aşarak sürekli thrashing ve yüksek TLB/önbellek kaçırma oranlarına yol açmaktadır.
- **NUMA ve Sanallaştırma Yükü:** Farklı NUMA düğümlerindeki belleğe erişim yavaşlığı ile sanallaştırmada kullanılan iç içe sayfa tabloları ek gecikme ve karmaşıklık getirmektedir [5].

Bu ciddi sorunlar, yüksek yoğunluklu çok çekirdekli sistemlerde sanal belleği iyileştirmeye yönelik kapsamlı araştırmaları beraberinde getirmiştir. Çözümler; yeni işletim sistemi sayfa değiştirme kurallarını, NUMA farkındalıklı bellek yönetimini ve otomatik sayfa göçünü kapsamaktadır. Bunlara ek olarak çok seviyeli TLB'ler, büyük sayfalar ve PMU güdümlü uyarlanabilir bellek politikaları gibi ortak tasarımlı özellikler ile NUMA'ya göre veri bölümleme gibi uygulama düzeyi stratejiler de öne çıkmaktadır [6-10].

Bu bölümün temel hedefi, güçlü çok çekirdekli sistemlerde sanal bellek optimizasyonuna ilişkin mevcut çalışmalara kapsamlı ve sistematik bir genel bakış sunmaktır. Konu alanı, bilgisayar mimarisi, işletim sistemleri ve uygulama performansının kesişiminde yer almaktadır.

II. SANAL BELLEK VE ÇOK ÇEKİRDEKLİ MİMARİLERE ARKA PLAN

Bu bölüm, sanal bellek optimizasyonu konusunu anlamak için gerekli temel kavramları açıklamaktadır. Sanal bellek üzerine kapsamlı bir kaynak için *Tanenbaum ve Bos [2]* ile *Silberschatz ve ark. [3]* başvurulabilir.

A. Sanal Bellek Temelleri

Sanal bellek, temel bir bilgisayar sistemi kavramıdır. Her programa gerçek fiziksel bellekten bağımsız kendi **özel sanal adres uzayını** kullanma imkânı sunar. İşletim sistemi ve donanım birlikte çalışarak bu sanal adresleri **çerçeve** adı verilen gerçek fiziksel bellek konumlarına eşler. Sanal uzay, sabit boyutlu **sayfalara**; fiziksel bellek ise aynı boyutlu **çerçevelere** bölünmüştür.

Sayfa-çerçeve eşlemesi, işletim sisteminin yönettiği **sayfa tablosuna** kaydedilir. Bir program bellekte bulunmayan ya da erişim iznine sahip olmadığı bir sayfaya erişmeye çalıştığında **sayfa hatası** tetiklenir ve çekirdek sorunu çözer. 64-bit sistemlerdeki devasa adres uzayları nedeniyle sayfa tabloları genellikle **hiyerarşik (çok seviyeli)** biçimde düzenlenmektedir.

Tablo 1. Sanal Belleğin Temel Kavramları ve Çok Çekirdekli Sistemlerdeki Önemi

Kavram	Tanım	Önemi
Sanal Adres Uzayı	Her işlemin kendine ait, tutarlı görünen geniş adres uzayı	İşlemler arası izolasyon ve korumanın temeli
Sayfa (Page)	Sanal belleğin sabit boyutlu birimi (tipik: 4 KB)	Bellek harita yönetiminin temel birimi
Çerçeve (Frame)	Fiziksel belleğin sabit boyutlu birimi	Sanal sayfaların fiziksel karşılığı
Sayfa Tablosu	Sayfa → çerçeve eşlemesini tutan OS veri yapısı	Adres çevirisinin temel kaynağı
TLB	Son kullanılan çevirileri saklayan hızlı donanım önbelleği	Sayfa tablosu erişiminin azaltılması
Sayfa Hatası	Erişilen sayfa bellekte olmadığında tetiklenen kesme	OS'nin gerektiğinde sayfaları belleğe yüklemesini sağlar

Kaynak: *Tanenbaum & Bos (2015); Silberschatz, Galvin & Gagne (2018); Denning (1968).*

Her bellek erişiminde sayfa tablosunu kontrol etmek son derece yavaş olacağından işlemciler, son kullanılan çevirileri depolayan küçük ve hızlı önbellekler olan Çeviri İzleme Arabelleklerini (TLB) kullanmaktadır. Bir adres istendiğinde donanım önce TLB'yi kontrol eder. Eğer çeviri bulunursa (**TLB** isabeti) fiziksel adrese hızla ulaşılır; bulunamazsa (TLB kaçırma**sı**) işlemci, çok seviyeli sayfa tablosunu bellekte arayan bir sayfa tablosu yürüyüşü gerçekleştirmelidir. Bu yürüyüş ek bellek erişimleri gerektirdiğinden sayfa tablosu verisi normal önbelleklerde yer almıyorsa önemli zaman kaybı doğurur. Bu nedenle TLB isabeti oranı, genel bellek erişim hızını doğrudan etkileyen kritik bir faktördür [1].

B. Önbellek Hiyerarşisi ve Tutarlılık

Sanal bellek sisteminin altında CPU ile DRAM arasındaki hız farkını kapatmaya yarayan önbellek hiyerarşisi (L1, L2, Son Seviye Önbellek) bulunmaktadır. Sayfa tablosu girişleri de bellekte depolanmakta ve uygulama verileriyle birlikte önbelleğe alınmaktadır; bu durum TLB kaçırmalarının

önbellekte trafik yaratmasına ve kullanışlı program verilerini dışlamasına neden olmaktadır.

Çok çekirdekli sistemlerde her çekirdek kendi özel önbelleklerine ve TLB'sine sahiptir. Bellek yapısı ise **tutarlılık protokolleriyle** (MESI gibi) veri bütünlüğü sağlanarak paylaşılmaktadır. Çok yüksek çekirdek sayılarını karşılamak üzere çoğu sistem **Tekdüze Olmayan Bellek Erişimi (NUMA)** mimarisi kullanmaktadır: fiziksel bellek **düğümlere** bölünmüş olup her düğümün kendi çekirdekleri ve yerel bellek denetleyicileri mevcuttur. Bir çekirdek kendi düğümündeki belleğe, uzak bir düğümden çok daha hızlı erişebilmektedir [6].

C. TLB Shootdown Mekanizması

İşletim sistemi bir bellek eşlemesini değiştirdiğinde (sayfayı kaldırma veya izinleri güncelleme), sayfa tablolarını ve TLB'leri senkronize tutmak için **geçersiz kılma** ya da **shootdown** işlemleri kullanılması zorunludur. Bu mekanizma birden fazla çekirdekte eski çevirilerin kullanılmaya devam etmesini engeller. Ancak büyük ölçekli sistemlerde TLB shootdown; IPI fırtınaları, gecikmeli senkronizasyon ve önemli performans kayıpları gibi ciddi zorluklar doğurabilmektedir.

III. YÜKSEK YOĞUNLUKLU ÇOK ÇEKİRDEKLİ SİSTEMLERDE PERFORMANS SORUNLARI

Çekirdek sayısı yükseldikçe sanal bellek sistemi, küçük sistemlerde neredeyse hissedilmeyen sorunlarla yüzleşmek zorunda kalmaktadır. Bu sorunlar; TLB ve sayfa tablosu bilgilerinin çok sayıda çekirdek arasında tutarlı tutulması gerekliliğinden, büyük ve çeşitli iş yüklerini yönetme zorunluluğundan, NUMA etkilerinden ve sanallaştırma katmanlarından kaynaklanmaktadır.

Tablo 2. Yüksek Yoğunluklu Çok Çekirdekli Sistemlerde Temel Sanal Bellek Performans Sorunları

Sorun	Mekanizma	Etki
TLB Shootdown	Eşleme değişiminde IPI mesajları ile tüm çekirdekler durdurulur	Gecikme artışı, çekirdek yük artışı
Sayfa Tablosu Çekişmesi	Çok iş parçacığı aynı tabloyu kilitler	Serileşme, önbellek hattı sıçraması
Yüksek Sayfa Hatası Oranı	Büyük çalışma kümeleri sayfa hatalarını artırır	Çekirdek işlem yükü, önbellek kirliliği
TLB Thrashing	Toplam çalışma kümesi TLB kapasitesini aşar	Sürekli TLB kaçırma, sayfa tablosu yürüyüşleri

NUMA Uzak Erişimi	Sayfa yanlış düğümde konumlandırılır	Yüksek erişim gecikmesi, bant genişliği baskısı
Sanallaştırma Maliyeti	İç içe sayfa tabloları ek çeviri katmanları ekler	TLB kaçırmalar daha pahalı hâle gelir

Kaynak: Majo & Gross (2011a); Gaud vd. (2014); Jia, Ding & Jiang (2023); Esmaeilzadeh, Burger & Austin (2013).

A. TLB Shootdown Yükü

TLB shootdown maliyeti, performans sorunlarının başında gelmektedir. İşletim sistemi bir eşlemeyi değiştirdiğinde, tüm çekirdekler eski TLB girişini silmek üzere durdurulmalıdır. Onlarca veya yüzlerce çekirdeğe sahip bir sistemde bu işlem, çok sayıda IPI mesajının iletilmesini ve makinenin büyük bölümünün kısa süreliğine durdurulmasını gerektirmektedir. Sanal makine veya NUMA dengeleyicisi gibi sürekli bellek tahsis eden veya taşıyan iş yükleri sık sık shootdown'a neden olabilmektedir.

B. Sayfa Tablosu Çekişmesi

Sayfa tabloları çakışma noktasına dönüşmektedir. Sayfa tabloları bir işleme özgü olsa bile o işlemi paylaşan birden fazla iş parçacığı eşzamanlı olarak sayfa hatalarına neden olabilir ya da eşlemeleri güncellemeye çalışabilir. Bu yapıları korumaya yönelik kilitler, çok sayıda iş parçacığının aynı kilitlere rekabet etmesiyle **serileşmeye** ve önbellek hattı sıçramasına yol açarak performansı olumsuz etkilemektedir [12].

C. Yüksek Sayfa Hatası Oranları ve Önbellek Thrashing

Grafik analizi gibi düzensiz veya büyük miktarda bellek kullanan iş yükleri, çekirdek sayısı arttıkça sayfa hatalarını yoğunlaştırmaktadır. Her hata; tablo araması ve çerçeve tahsisi gibi çekirdek işlemlerini tetikler. Tüm bu işlemler CPU önbelleklerini doldurarak uygulamanın kullanışlı verilerini dışlar. Yüksek eşzamanlılıkta sayfa hatası işleme kritik bir darboğaza dönüşebilir.

Çalışan tüm iş parçacıklarının toplam bellek gereksinimi hem özel hem de paylaşılan önbelleklerin kapasitesini sıklıkla aştığından TLB'ler de hızla dolmakta; bu durum yüksek TLB kaçırma oranlarına, daha fazla sayfa tablosu yürüyüşüne ve ek bellek trafiğine yol açmaktadır.

D. NUMA Karmaşıklığı

Bellek farklı erişim hızlarına sahip düğümlere bölündüğünden sayfanın fiziksel konumu kritik önem taşımaktadır. İşletim sistemi bir sayfa için kötü bir konum seçerse çok sayıda yavaş uzak bellek erişimi yaşanabilir ve ara bağlantılar aşırı yüklenebilir. Bu sorunu gidermek için kullanılan **sayfa göçü**, veri

kopyalama ve TLB shootdown maliyetleri nedeniyle kendi başına önemli bir ek yük oluşturmaktadır. Bellek yöneticisinin hem erişim hızını hem de göç maliyetlerini dengelemesi gerekmektedir [9].

E. Sanallaştırma Maliyetleri

Sanallaştırma, adres çevirisine ek bir katman eklemektedir: konuk sanal adresten konuk fiziksel adrese, oradan da sunucu fiziksel adresine dönüşüm gerekmektedir. Donanım desteğine (EPT/NPT gibi iç içe sayfa tabloları) rağmen TLB kaçırmalar daha pahalı hâle gelir; çünkü sistem daha fazla çeviri seviyesini kontrol etmek zorundadır. VM göçü veya bellek aşırı taahhüdü gibi işlemler eşleme değişikliklerini sıklaştırarak hem sunucu hem de konuk tarafında sanal bellek alt sistemine ek yük bindirmektedir [11].

IV. SAYFA DEĞİŞTİRME VE ÖNBELLEKLEME POLİTİKALARI

Sayfa değiştirme politikaları; bellek kıtlığında hangi sayfaların fiziksel bellekte kalacağına, hangilerinin çıkarılacağına karar vermektedir. Çok çekirdekli sistemlerde bu kararlar önbellekler, TLB'ler ve NUMA yerleşimi ile doğrudan bağlantılıdır.

Tablo 3. Sayfa Değiştirme Algoritmalarının Karşılaştırması

Algoritma	Çalışma Prensibi	Avantajlar	Dezavantajlar
FIFO	İlk giren sayfayı çıkar	Basit uygulama, düşük ek yük	Belady anomalisine açık
LRU	En uzun süre kullanılmayanı çıkar	İyi yaklaşık performans, teorik dayanağı güçlü	Tam izleme yüksek maliyet gerektirir
CLOCK	LRU yaklaşımı; bit ve dairesel liste kullanır	Pratik, düşük maliyet, LRU benzeri	Çok çekirdekli ortamda kilit çekişmesi
ARC	Sık + son kullanılanlar için iki liste tutar	Değişen erişim desenlerine uyar	Karmaşık uygulama; patent sorunu mevcuttu
Uyarlanabilir/ML	PMU verisine veya makine öğrenmesine dayalı	Dinamik iş yüklerine uyum, düşük çöpe çıkarma	Yüksek ek yük olasılığı, mühendislik karmaşıklığı

Kaynak: Belady (1966); Denning (1968); Tanenbaum & Bos (2015).

A. Geleneksel Algoritmalar

İlk VM algoritmaları arasında FIFO (İlk Giren İlk Çıkar), LRU (En Az Son Kullanılan) ve CLOCK yer almaktadır. LRU; kısa süre önce kullanılan sayfaların yakın gelecekte de kullanılacağını varsayarak en eski sayfayı çıkarmaktadır.

Ancak tam LRU izleme çok maliyetli olduğundan işletim sistemleri, sayfaya ikinci bir şans tanıyan CLOCK gibi basitleştirilmiş yaklaşımlar kullanmaktadır [1, 4]. CLOCK verimli olsa da çok çekirdekli ortamlar için büyük ölçüde tasarlanmamıştır: birçok uygulama listelerini kilitlerle korumakta ve bu durum yüksek çekirdekli sistemlerde kilit çekişmesine ve önbellek hattı sıçramasına yol açmaktadır.

B. Çalışma Kümeleri ve Önbelleklemeyle Etkileşim

Belady'nin 1966'daki çalışması [4], teorik en uygun algoritmanın gelecekteki sayfa kullanımını önceden bilmesi gerektiğini göstermiştir; bu durum pratikte uygulanamaz olsa da araştırmalar için önemli bir referans olmaya devam etmektedir. Çalışma kümesine dayalı yöntemler de ölçeklenmekte güçlük çekmektedir: çok sayıda iş parçacığının eşzamanlı çalıştığı durumlarda toplam çalışma kümesi bellek kapasitesini aşabilmekte ve thrashing'e neden olmaktadır.

Değiştirme politikaları ayrıca uygulama belleği (anonim bellek) ile sayfa önbelleği (dosya verileri) arasındaki dengeyi de yönetmelidir. Dosya önbelleği sayfalarını agresif biçimde tutmak uygulama bellek hatalarına, tam tersi ise dosya erişim performansının düşmesine neden olabilir. Ön okuma (prefetching) hatalı tahminler sonucu gereksiz veri yüklenerek çalışma kümesi sayfalarının yerinden edilmesine yol açabileceğinden bu dengeyi daha da karmaşıklaştırmaktadır.

C. Uyarlanabilir ve Makine Öğrenmesi Tabanlı Politikalar

Güncel araştırmalar uyarlanabilir ve makine öğrenmesi (ML) tabanlı politikaları önermektedir. Uyarlanabilir politikalar; hata oranları ve TLB davranışı gibi metrikleri izleyerek parametrelerini dinamik olarak ayarlamaktadır. ML tabanlı sistemler ise çıkarmayı bir tahmin problemi olarak ele alıp geçmişe ve performans verilerine dayanarak hangi sayfaların en az kullanılacağını tahmin etmektedir. Bu akıllı sistemlerin aşırı yük getirmemesi için dikkatli bir mühendislik yaklaşımı gerekmektedir.

V. NUMA BİLİNÇLİ BELLEK YÖNETİMİ VE VERİ YERLEŞİMİ

NUMA mimarileri, yüksek yoğunluklu çok çekirdekli sistemlerde artık standarttır. Yerel bellek erişimi uzaktan erişimden çok daha hızlı ve verimli olduğundan sayfaların fiziksel yerleşimi son derece büyük önem taşımaktadır [6, 9].

Tablo 4. NUMA Bellek Yerleştirme Stratejilerinin Karşılaştırması

Strateji	Açıklama	En Uygun Durum	Olumsuz Yönler
İlk Dokunuş Tahsisi	Sayfa, ilk erişilen düğüme yerleştirilir	İş parçacığı başına veri bölünmesi	Paylaşılan/taşınan veriler için yetersiz
Döngüsel Dağıtım	Sayfalar düğümler arasında sırayla dağıtılır	Bant genişliği dengesinin önemli olduğu durum	Ortalama gecikmeyi artırabilir
Açık NUMA API'leri	Geliştirici numactl vb. ile yerleştirmeyi kontrol eder	Uzman optimizasyonu gerektiren iş yükleri	Manuel ayarlama; kod taşınabilirliği düşer
Otomatik Sayfa Geçişi	OS, PMU verisiyle sayfaları aktif olarak taşır	Erişim deseni değişen dinamik iş yükleri	Göç başına kopyalama + TLB shootdown maliyeti
Sayfa Renklendirme	Çerçeve seçimiyle önbellek kümesi çakışmaları azaltılır	Önbellek çakışmasına duyarlı iş yükleri	Nüanslı uygulama; bazı platformlarda etki sınırlı

Kaynak: Majo & Gross (2011a); Lepers, Quema & Fedorova (2015); Pan, Ma, Liu & Gu (2021).

A. Yerleşim Stratejileri

İşletim sistemleri birkaç temel yerleşim politikası kullanmaktadır:

- **İlk Dokunuş Tahsisi:** Bellek, sayfaya ilk kez erişilen düğüme yerleştirilir. İş parçacıkları kendi veri bölümlerine bağlı kalırsa iyi sonuç verir; ancak iş parçacıkları yer değiştirirse veya veriler geniş çapta paylaşılıyorsa yanlış yerleşime, dolayısıyla yavaş uzak erişimlere yol açabilir [6].

- **Döngüsel Dağıtım:** Sayfalar düğümler arasında sırayla dağıtılarak bellek bant genişliği ve kapasitesi dengelenir. Dezavantajı, bir düğümdeki iş parçacıklarının sürekli uzak belleğe erişerek ortalama gecikmeyi artırabilmesidir.

- **Açık NUMA API'leri:** numactl gibi araçlarla geliştiricilerin iş parçacığı ve bellek yerleşimini manuel olarak belirlemesine olanak tanır. Belirgin performans kazanımları sağlayabilir; ancak karmaşık manuel ayarlama gerektirmektedir.

B. Dinamik Sayfa Göçü

İş yükleri değiştiğinden statik yerleşim yeterli olmayabilmektedir. NUMA bilinçli yöneticiler, sistem çalışırken yerleşimi düzenlemek için **sayfa göçü** kullanmaktadır. Yerel veya uzak erişimleri izleyerek işletim sistemi, sık erişilen sayfaları kullanan çekirdeklere yaklaştırabilmektedir. Göç; veri kopyalamayı, sayfa tablosunu güncellemeyi ve TLB shootdown gerçekleştirmeyi içermektedir.

Doğru kullanıldığında uzak erişim sayısını azaltır; aşırı kullanıldığında ise uzak erişim sorunlarını göç ek yükleriyle takas etmekten ibaret kalır [9].

C. Bellek Sıkıştırma ve Sayfa Renklendirme

Bellek sıkıştırma (Memory Compaction), parçalanmış fiziksel belleği yeniden düzenleyerek büyük sayfaların tahsis edilebileceği sürekli alanlar oluşturmaktadır; ancak bu işlem zaman almakta ve sayfaların taşınmasını gerektirmektedir. Öte yandan sanal bellek, **sayfa renklendirme** aracılığıyla belirli fiziksel çerçeveler seçilerek sayfaların önbellek kümelerine veya DRAM bankalarına eşlenme biçimini etkileyebilmektedir. Bu teknik, paylaşılan önbelleklerdeki gürültüyü azaltmakta ve bellek trafiğini farklı banklara yayarak performansa katkı sağlamaktadır [10].

VI. SANAL BELLEK OPTİMİZASYONU İÇİN DONANIM-YAZILIM ORTAK TASARIMI

Sanal bellek performansı, donanıma yerleştirilmiş özellikler ile yazılımın uyguladığı politikalar arasındaki sıkı işbirliğine dayanmaktadır. Bu bölüm, etkili VM optimizasyonu için donanım ve yazılımın birlikte nasıl tasarlandığını incelemektedir.

Tablo 5. Donanım-Yazılım Ortak Tasarım Tekniklerinin Karşılaştırması

Teknik	Mekanizma	Performans Faydası	Önemli Notlar
Büyük Sayfalar	2 MB / 1 GB sayfalar; her TLB girişi daha büyük alanı kapsar	Daha az TLB kaçırma, daha az PT yürüyüşü	İç parçalanma; NUMA'da dikkatli kullanım gerekir
Şeffaf Büyük Sayfalar (THP)	OS belleği otomatik olarak büyük sayfalar halinde birleştirir	Geliştirici müdahalesi olmaksızın TLB iyileştirmesi	Sayfa birleştirme gecikmesi; bazı iş yüklerinde zararlı
Çok Seviyeli TLB	Hızlı L1 TLB; daha büyük L2 TLB ile desteklenir	Kapasiteyi artırırken erişim gecikmesini düşük tutar	Donanım alanı ödünleşmesi
ASID Etiketleme	TLB girişleri işlem kimliğiyle etiketlenir	Bağlam geçişinde tam TLB temizleme önlenir	ASID alanı; TLB takma adı karmaşıklığı
İç İçe Sayfa Tabloları (EPT/NPT)	MMU, konuk ve sunucu PT çevirilerini donanımda yönetir	Tuzakları azaltır; hipervizörü basitleştirir	TLB kaçırmalar daha fazla çeviri seviyesi

			nedeniyle pahalılaşır
PMU Tabanlı Politikalar	TLB kaçırma/uzak erişim oranları politika kararlarını yönlendirir	Uyarlanabilir THP/göç politikaları; koşul bazlı iyileştirme	Örnekleme ek yükü; dikkatli politika tasarımı gerekli

Kaynak: Gaud vd. (2014); Choi vd. (2019); Jia, Ding & Jiang (2023); Merkel, Stoess & Bellosa (2010); Esmaeilzadeh, Burger & Austin (2013).

A. Büyük Sayfalar ve Şeffaf Büyük Sayfalar

Çoğu mimari birden fazla sayfa boyutunu (ör. 4 KB, 2 MB, 1 GB) desteklemektedir. Büyük sayfaların en önemli yararı TLB kaçırma oranını düşürmesidir: tek bir büyük TLB girişi çok daha geniş bir bellek alanını kapladığından daha az giriş yeterli olmakta ve sayfa tablosu yürüyüşleri azalmaktadır. İşletim sisteminin bu büyük sayfaları ne zaman kullanacağına, ne zaman küçük sayfalara geri döneceğine karar vermesi gerekmektedir. Bazı sistemler bunu Şeffaf Büyük Sayfalar (THP) ile otomatikleştirmektedir [7, 8].

Büyük sayfalar TLB açısından verimli olsa da esnekliği azaltmaktadır: büyük bir sayfa yalnızca kısmen kullanılıyorsa kalan bellek boşa gitmektedir (iç parçalanma). Ayrıca büyük sayfalar, bazı NUMA sistemlerinde dikkatli kullanılmadığı takdirde performansı olumsuz etkileyebilmektedir [7].

B. Gelişmiş TLB Mimarileri

TLB'ler giderek daha karmaşık bir yapı kazanmaktadır. Günümüzde çok seviyeli TLB hiyerarşileri (hızlı L1 ve daha büyük L2) kullanılmakta, bazı tasarımlarda veri paylaşımını kolaylaştırmak için çekirdek gruplarına paylaşımlı TLB'ler uygulanmakta; diğer durumlarda ise çakışmaları önlemek amacıyla çekirdek başına TLB tercih edilmektedir. Adres Uzayı Tanımlayıcıları (ASID) gibi özellikler, TLB girişlerini işlem kimliğiyle etiketleyerek bağlam geçişlerinde TLB'nin tümüyle temizlenmesine gerek kalmaksızın çoklu işlem desteği sağlamaktadır.

C. Sanallaştırma Donanım Desteği

Donanım artık sanallaştırmanın gerektirdiği iki katmanlı çeviriyi (iç içe sayfa tabloları: Intel EPT ve AMD NPT) yönetmeye yardımcı olmaktadır. Bu destek, MMU'nun konuk ve sunucu sayfa tablolarını donanım düzeyinde yönetmesine olanak tanıyarak hipervizörü basitleştirmektedir [11]. Bedeliyse TLB kaçırmaların daha pahalı hâle gelmesidir; çünkü sistem daha fazla çeviri seviyesini kontrol etmek zorundadır. Hipervizörler ve donanım bu sorunu birlikte

ele almaktadır: konuk fiziksel belleği için büyük sayfalar kullanma ve sayfa tablosu güncellemelerini optimize etme gibi teknikler NUMA bilinçli VM yerleşimiyle desteklenmektedir.

D. PMU Güdümlü Uyarlanabilir Politikalar

Modern işlemciler TLB kaçırmaları, sayfa tablosu yürüyüşleri ve yerel/uzak bellek erişim oranları gibi olayları sayan Performans İzleme Birimlerine (PMU) sahiptir. İşletim sistemi veya hipervizör bu PMU verilerini zekice ve uyarlanabilir kararlar almak için kullanabilmektedir; ör. TLB kaçırma oranlarına göre THP'nin ne zaman kullanılacağını belirlemek veya uzak erişim oranları çok yükseldiğinde sayfa göçünü başlatmak. Bu örnekler, VM performansını optimize etmenin donanım ile yazılım arasında eşgüdümlü çaba gerektirdiğini açıkça ortaya koymaktadır [13, 14, 15].

VII. VAKA ÇALIŞMALARI

Bu bölüm, sanal bellek optimizasyon tekniklerinin gerçek dünyadaki sistemlere nasıl uygulandığını somut örnekler üzerinden incelemektedir.

Tablo 6. Gerçek Sistemlerde Sanal Bellek Optimizasyonu: Vaka Çalışmaları Özeti

Platform	Temel VM Sorunları	Uygulanan Optimizasyonlar	Gözlemlenen Kazanımlar
Linux Çekirdeği	TLB shootdown ölçeklenmesi, NUMA yerleşimi, bellek sıkıştırma	THP, NUMA Dengeleme (AutoNUMA), mmap_sem kilidi bölünmesi, ksmbd	NUMA iş yüklerinde belirgin gecikme azalması; büyük sayfalarla TLB kaçırmaların azaltılması
Büyük Veri (Spark/Hadoop)	Karışık erişimli büyük çalışma kümeleri, NUMA uzaklıkları	Açık büyük sayfa tahsisi, NUMA bilinçli iş parçacığı yerleşimi, bellek önceden alma	Sorgu süresinde %10-30 iyileştirme; bellek bant genişliği darboğazlarının azalması
Bulut Sanallaştırma (KVM/QEMU)	İç içe PT maliyetleri, aşırı bellek, VM göçü	EPT büyük sayfaları, KSM (çekirdek aynı sayfa birleştirme), NUMA bilinçli VM yerleşimi	Sanallaştırma yükünün azaltılması; sanal makinelerde daha yüksek yoğunluk

Kaynak: Tanenbaum & Bos (2015); Majo & Gross (2011a, 2011b); Jia, Ding & Jiang (2023); Operating Systems Group, Hamburg University of Technology (2020).

A. Linux Çekirdeği

Linux çekirdeği, yıllar içinde VM performansını iyileştirmek için pek çok mekanizma kazanmıştır. THP (Şeffaf Büyük Sayfalar), bellekte sürekli bölgeler oluşturarak otomatik biçimde büyük sayfa tahsis etmekte ve TLB kaçırmaları önemli ölçüde azaltmaktadır. Çekirdeğin AutoNUMA özelliği sayfa erişimlerini örnekleyerek göçü otomatik olarak tetiklemektedir. Çekirdeğin sonraki sürümlerinde ise mmap_sem kilidinin bölünmesi ve RCU tabanlı güncelleme yolları gibi çeşitli kilitleme iyileştirmeleriyle sayfa tablosu çekişmesi ciddi ölçüde azaltılmıştır [2].

B. Büyük Veri İş Yükleri (Spark/Hadoop)

Apache Spark ve Hadoop çerçeveleri, oldukça büyük ve düzensiz bellek erişim desenleri oluşturmaktadır. Bu ortamlarda açık büyük sayfa tahsisi ile NUMA bilinçli iş parçacığı yerleşimi uygulamaları sorgu sürelerinde %10-30 arasında iyileştirme sağlamıştır. Bellek önceden alma ve prefetching teknikleri ise gecikmeyi gizlemeye ve bant genişliği darboğazlarını azaltmaya yardımcı olmaktadır [6].

C. Bulut Sanallaştırma (KVM/QEMU)

KVM/QEMU gibi hipervizör platformları birden fazla sanal makineyi çalıştırdığından bellek yönetimi son derece karmaşık hâle gelmektedir. EPT büyük sayfaları iç içe TLB kaçırma maliyetlerini azaltmaktadır. KSM (Çekirdek Aynı Sayfa Birleştirme), VM'ler arasındaki özdeş sayfaları şeffaf biçimde birleştirerek bellek kullanımını düşürmektedir. NUMA bilinçli VM yerleşimi ise konuk bellek alanının fiziksel NUMA topolojisiyle hizalanmasını sağlamakta ve gereksiz uzak erişimleri önlemektedir [11].

VIII. SONUÇ VE AÇIK ARAŞTIRMA SORULARI

Yüksek yoğunluklu çok çekirdekli sistemler, artan TLB shootdown yükleri, sayfa tablosu çekişmesi, NUMA karmaşıklığı ve sanallaştırma maliyetleri nedeniyle geleneksel sanal bellek altyapısına yönelik ciddi sorunlar oluşturmaktadır. Bu bölümde ele alınan teknikler üç geniş kategori altında toplanabilir:

- **Yazılım düzeyindeki çözümler:** Uyarlanabilir sayfa değiştirme algoritmaları, AutoNUMA gibi NUMA bilinçli bellek yönetimi, kilitsiz veri yapıları ve geliştirilmiş page-cache yönetimi.
- **Donanım özellikleri:** Büyük sayfalar, çok seviyeli TLB'ler, ASID etiketleme, EPT/NPT ve PMU olay sayaçları.

- **Ortak tasarım teknikleri:** PMU güdümlü THP kullanımı, NUMA bilinçli sayfa göçü, dinamik bellek sıkıştırma ve sanallaştırma farkındalıklı sayfa tablosu optimizasyonu.

Birkaç araştırma sorusu hâlâ açık kalmaktadır:

- **Uyarlanabilir VM:** Makine öğrenmesi tabanlı sayfa değiştirme politikaları kabul edilebilir ek yükle farklı iş yüklerine pratikte genellenebilir mi?
- **Heterojen Sistemler:** GPU ve işlemci belleğini bütünleşik sanal adres uzayıyla birleştiren yeni mimarilerde VM optimizasyonu nasıl ele alınacaktır?
- **Güvenlik Kısıtlamaları:** Meltdown/Spectre saldırıları nedeniyle uygulanan adres-uzayı izolasyon yamaları (ör. KPTI) TLB maliyetlerini artırmaktadır; güvenlik ve VM performansı nasıl uzlaştırılacaktır?
- **Kalıcı Bellek (NVM):** Intel Optane gibi kalıcı bellek birimleri yeni NUMA katmanları oluşturmaktadır; VM sistemi bu heterojen bellek hiyerarşisine nasıl uyarlanacaktır?

Genel olarak değerlendirildiğinde, tüm teknoloji yığını boyunca kapsamlı optimizasyon gerektiren ve birden fazla katmanda eşgüdümlü çabayı zorunlu kılan VM performansı, modern bilgisayar sistemleri araştırmalarında merkezi bir yer tutmaya devam edecektir.

Kaynakça

Belady, L. A. (1966). A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal*, *5*(2), 78–101.

Bryant, R. E., & O'Hallaron, D. R. (2016). *Computer systems: A programmer's perspective* (3. bs.). Pearson.

Choi, G., Son, J., Choi, J., Cho, S.-J., & Won, Y. (2019). HPanal: A framework for analyzing tradeoffs of huge pages. *Proceedings of the ACM/SIGAPP Symposium on Applied Computing (SAC)*, 1438–1443.

Denning, P. J. (1968). The working set model for program behavior. *Communications of the ACM*, *11*(5), 323–333.

Esmaeilzadeh, H., Burger, D., & Austin, T. (2013). Future scaling of memory systems: Challenges for virtual memory in many-core architectures. *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 125–136.

Gaud, F., Lepers, B., Decouchant, J., Funston, J., Fedorova, A., & Quema, V. (2014). Large pages may be harmful on NUMA systems. *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*, 231–242.

Jia, W., Ding, X., & Jiang, X. (2023). HugeGPT: Storing guest page tables on host huge pages for efficient nested paging. *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 1–14.

Lepers, B., Quema, V., & Fedorova, A. (2015). Thread and memory placement on NUMA systems: Asymmetry matters. *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*, 277–289.

Majo, Z., & Gross, T. R. (2011a). Memory management in NUMA multicore systems: Trapped between cache contention and interconnect overhead. *Proceedings of the International Symposium on Memory Management (ISMM)*, 11–20.

Majo, Z., & Gross, T. R. (2011b). Memory system performance in a NUMA multicore multiprocessor. *Proceedings of the ACM International Conference on Systems and Storage (SYSTOR)*, 1–12.

Merkel, A., Stoess, J., & Bellosa, F. (2010). Resource-conscious scheduling for energy efficiency on multicore processors. *Proceedings of the European Conference on Computer Systems (EuroSys)*, 153–166.

Operating Systems Group, Hamburg University of Technology. (2020). *Virtual memory on NUMA systems* [Ders notu]. Hamburg University of Technology.

Pan, X., Ma, J., Liu, Y., & Gu, Z. (2021). NUMA-aware memory coloring for multicore real-time systems. *Journal of Systems Architecture*, *115*, 102–115.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating system concepts* (10. bs.). Wiley.

Tanenbaum, A. S., & Bos, H. (2015). *Modern operating systems* (4. bs.). Pearson.