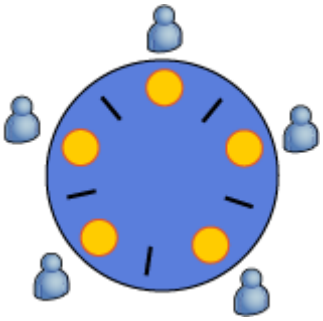
İşletim Sistemleri

Hasan Bank

4 Aralık 2015

1 Filozofların Akşam Yemeği

1.Tanımı
Yuvarlak bir masa etrafındaki n adet(n≥2) filozofun yemek yeme problemidir.Yemek olarak iki çatalla yenilen spagetti düşünülebilir ancak şekilde de görüldüğü gibi her bir filozofun iki yanında 1'er çatal bulunmaktadır.[1]



2.Açıklama
Her bir filozof sağındaki çatalı alırsa hiç biri yemek yiyemeyecektir ve DeadLock oluşacaktır.Şayet , filozoflar her iki çatalı da almak için uğraşırlarsa racing conditions ortaya çıkar.Eğer her bir filozof diğeri yemek yesin diye çatal almaz ise Starvation oluşur.
3.Kim Tarafından Bulundu?
Bu problem , bilim dünyasına Dijkstra aracılığla katılmıştır.
4.Neyi Çözmeyi Hedefliyor?
Eş zamanlı işlem yönetimine hedefler.Bu amaç doğrultusunda , kısıtlı kaynağa problemsiz erişimi açıklamaya çalışır.
5.Kullanılan Çözüm Yöntemleri[2]
5.1 Rastegele Süre Çözümü (Random Solution)
Adından da anlaşılacağı üzere bu çözümde rastegele bir süre belirlenir.Filozoflar ilk çatalı aldıktan sonra eğer ikinci çatalı alabilirlerse yemeği yerler.Eğer alamazlarsa bu rastegele süre içinde çatalı beklerler ve yine çatalı alamazlarsa ellerinde tek çatalı da diğer filozoflar için masaya bırakırlar.
Rastgelelik hakim olduğu için problemi çözeceğinin garantisi yoktur.
5.2 Garson Çözümü (Conductor Solution)
Yukaraki sorunu garson aracılığı ile çözmek hedeflenmiştir.Bu amaç doğrultusunda garsonun işaretlediği filozof çatalları alır.Böylece işaretlenen yemek yerken sonraki yemez bir sonraki yer.Garson sırasıyla dönerek herkesin yemek yemesini

sağlar.

5.3 Monitör Çözücümü (Monitor Solution)

Önceki garson çözümüne benzemektedir.Filozoflar kendinden öncekine göre yemek yer ya da yemez.Aynı garson örneğinde olduğu gibi sıra ile yemek yenir.Örneğin önce 1,3,5 nolu filozoflar yemek yerken daha sonra 2,4,6 nolu filozoflar yemek yer.

5.4 Chandy Misra Çözümü (Chandy Misra Solution)

Öncelikle merkezi kontrol sistemi devre dışı bırakışmıştır.Bu çözümün önemli iki noktasından birisi çatalların temiz yada kirli olmasıdır.Masadaki çatallar temizdir.Filozof eğer yemek yerse çatal kirlenir.Eğer bir filozof yanındaki fiozofun çatalını isterse bu çatalın kirli/temiz durumuna bakılır.Kirli ise yemek yemiş demektir ve yanındaki filozofa vermek zorundadır.Temiz ise daha yememiştir ve çatalı vermez. Problemin açıklamasında bahsettiğim DeadLock'a düşmemek için de bu yöntemin en önemli ikinci özelliği olan filozoflara öncelik atama kullanılır.Böylece, filozoflar önceliklerine göre çatalları alacağından DeadLock engellenmiş olur.

2 Banker Algoritması

1.Tanımı

Banker Algoritması kaynak dağılımını gerçekleştirip deadlock olayını engellemek için geliştirilmiş bir algoritmadır.[3]

2.Açıklama

Yukarıda bahdesilen kaynak yönetimin gerçeklenmesi için bazı veriler kullanılır.Bunlar:
-Her işlemin ihtiyaç duydyuğu kaynak miktarı
-Her işlemin şu an ne kadar kaynak tuttuğu
-Şu anda ne kadar kaynağın erişlebilir
olduğudur.[4] Eğer ihtiyaç duyulan kaynak azami kaynaktan fazla ise işleme izin verilemez.Şayet azami kaynaktan fazla değil fakat eldeki kaynaktan fazla ise işlem beklemelidir.Algoritma bu yöntemleri kullanarak işlemin deadlokca düşlmesini engellemeye çalışır. "Safe State" ve "UnSafe State" kavramlarına çözüm yöntemleri başlığı altında değinildi.

3.Kim Tarafından Bulundu?

Algoritma Dijkstra tarafından geliştirilmiştir.

4.Neyi Çözmeyi Hedefliyor?

Tanımda da söylendiği üzere algoritma kaynak dağılımını yaparak sistemin deadlock'a düşmesini çözmeyi hedefliyor.

5.Kullanılan Çözüm Yöntemleri

Burada bir örnek ile açıklama yapılabilir.

Sistemin sahip olduğu kaynak miktarı

A	B	C	D
3	1	1	2

İşlemlerin şu anda sahip olduğu kaynaklar

X	A	B	C	D
P1	1	2	2	1
P2	1	0	3	3
P3	1	1	1	0

İşlemlerin maximum kullanabileceği kaynak miktarı

X	A	B	C	D
P1	3	3	2	2
P2	1	2	3	4
P3	1	1	5	0

-P1 2 A,1 B ve 1 D aldıktan sonra maximuma ulaşır ve sonlanır.P1 sahip olduğu tüm kaynakları sisteme devreder ve sistem 4 A,3 B,3 C,3 D'ye sahip olur.
-P2 2B ve 1 D alır ve sonlanır.Sistemde 5 A,3 B,6 C ve 6 D olur.
-Son olarak P3 4 C alır ve sonlanır.
Tüm işlemler sonlandığı için bu duruma "safe" denir.[5]
Ancak , eğer 2.işlem başlangıçta B kaynağından 2 adet almak isterse sistem bunu sağlayamayacağı için "unsafe" durum oluşur.
"Unsafe" durumlar deaclock'a yol açabilir fakat kesin değildir.2.işlemin herkesten önce başlayıp ve de maximum ihtiyacını karşılaması sadece bir olasılıktır.Bu yüzden, her "unsafe" durum deadlock oluşturur diyemeğiz.Bu sadece olasılıklardan biridir.[6]

References

[1] İlke Demir. Ölümcül kilitlenme. http://e-bergi.com/y/deadlock.

[2] Şadi Evren ŞEKER. Filozofların akşam yemeği (dining philosophers). http://bilgisayarkavramlari.sadievrenseker.com/2012/01/22/filozoflarin-aksam-yemegi-dining-philosophers/.

[3] idc online.com. Banker's algorithm. http://www.idc-online.com/technical_references/pdfs/information_technology/Banker_Algorithm.pdf.

[4] Şadi Evren ŞEKER. Banker algoritması (banker's algorithm). http://bilgisayarkavramlari.sadievrenseker.com/2011/06/26/banker-algoritmasi-bankers-algorithm/.

[5] classle.net. Bankers algorithm. https://www.classle.net/book/bankers-algorithm.

[6] Wandering Logic. Why unsafe state not always cause deadlock? http://cs.stackexchange.com/questions/45145/why-unsafe-state-not-always-cause-deadlock.