ÖZET

DAĞITIK UYGULAMALAR ARASI SENKRONİZASYON YÖNTEMİ

Buluş, dağıtık şekilde aynı veya farklı frekansta çalışan simülatör uygulamalarını eş zamanlayarak uygulamaların birlikte hareket etmesini sağlayan bir yöntem ile ilgilidir.

İSTEMLER

1. Buluş, dağıtık şekilde aynı veya farklı frekansta çalışan simülatör uygulamalarını eş zamanlayarak uygulamaların birlikte hareket etmesini sağlayan bir yöntem olup;

- Eş zamanlı sunucu uygulamasının (3), farklı frekansta çalışacak uygulamalar (2) için senkron frekanslarını belirlemesi,
- Belirlenen her senkron frekansı için ayrı yüksek öncelikli iş parçacıklarının oluşturulması,
- Her iş parçacığının işletim sistemi çağrıları ile aldığı yüksek çözünürlüklü süreölçer ile veya kendisi için atanan seri porttan sinyal geldiğinde UDP protokolü kullanarak belirlenen frekansta çok noktaya yayın yapması,
- Eş zamanlı sunucu uygulamasının (3); süre hassasiyeti için iş parçacıklarını uyutmama veya küçük uyku süreleri ile daha düşük işlemci kullanımı seçeneğinin olması,
- Yayınlanan mesajın içerisinde senkron frekansı ve döngü sayacı bulunması,
- Uygulamalar (2) içerisinde yer alan eş zamanlama ajanlarının (4) döngünün bitiminde bariyeri kilitlemesi ve sunucu uygulaması tarafından yayınlanan mesajın gelmesini beklemesi,
- Gelen mesajın içerisindeki frekans ve döngü sayacının kontrol edilmesi,
- Bu iki bilginin tutarlı olduğu durumda bariyerin açılması ve sonraki adıma geçilmesi,
- Eğer uygulama kritiklik seviyelendirmesi yapıldıysa, eş zamanlama sunucu uygulamasında (3) yer alan iş parçacıklarının bir sonraki mesajı yayınlamak için uygulamalardan " döngü adımını tamamladım" mesajını beklemesi,
- Kritik uygulamaların belirlenmesi,
- Eş zamanlama sunucu uygulamasının (3) gelen mesajın içerisindeki uygulama kimliğinin kritik bir uygulamaya ait olup olmadığını ve döngü

sayacının ilgili iş parçacığından yayınlanan son senkron mesajı içerisindeki döngü sayacı ile eşitliğini kontrol etmesi

- Senkron periyodu tamamlandığında kritik uygulamaların (2) hepsinden tamamladım mesajı alındığı zaman, uygulamaların içerisindeki eş zamanlama ajanları (4) için devam mesajı yayınlanması,

adımlarını içermesi ile karakterize edilmektedir.

TARİFNAME

DAĞITIK UYGULAMALAR ARASI SENKRONİZASYON YÖNTEMİ

Teknik Alan

Buluş, dağıtık şekilde aynı veya farklı frekansta çalışan simülatör uygulamalarını eş zamanlayarak uygulamaların birlikte hareket etmesini sağlayan bir yöntem ile ilgilidir.

Önceki Teknik

Teknikte mevcut olan bir çok simülasyon sistemlerinde, çok sayıda uygulamanın eşzamanlı olarak çalışması zorunlu olmaktadır. Çok sayıda uygulamanın aynı anda çalışması, uygulamalar arasında senkronizasyon sorununu da beraberinde getirmektedir. Eğer uygulamalar arasında senkron uyumu olmazsa, simülatörlerde hatalar meydana gelmektedir.

Mevcut simülatör sistemlerinde senkronizasyon için harici bir donanıma ihtiyaç duyulmaktadır. Farklı frekansta çalışan uygulamalar için ise bir çözüm bulunamamaktadır.

Tekniğin bilinen durumunda yer alan CN111324046A sayılı dokümanda dağıtılmış bir simülasyon sistemi üzerinde ortak çalışmayı gerçekleştirmek için bir yöntem ve bir sistem açıklanmaktadır. İlgili yöntem, dağıtılmış simülasyon sisteminin içerdiği bir zaman senkronizasyon yöneticisinin tüm saat periyotlarını birleştirmek için skaler mantık zamanını kullanması, dağıtılmış simülasyon sistemindeki alt sistemlerin, dağıtılmış simülasyon sistemindeki olay frekansı beklentisine dayalı olarak zaman senkronizasyon yöneticisinin senkron saat frekansını ayarlanması; bir zaman gecikmesi hesaplama sonucu elde etmek için dağıtılmış simülasyon sisteminde simülasyon olayı iletiminin zaman gecikmesinin hesaplanması; ve dağıtılmış simülasyon sistemindeki her bir alt sistem üzerinde ortak çalışmayı

yürütmek için zaman gecikmesi hesaplama sonucuna göre simülasyon olayının bir varış zamanı dizisinin kontrol edilmesi, adımlarını içermektedir.

Tekniğin bilinen durumunda yer alan CN111147284A sayılı dokümanda verileri merkez olarak alan bir dağıtılmış gerçek zamanlı simülasyon sistemi için veri etkileşim stratejisi açıklanmaktadır. Verileri merkez olarak alan bir veri değişim mimarisi esas olarak tasarlanmıştır ve uygulanabilir bir veri protokolü standardı seçme, bir simülasyon tahrik saati senkronizasyon mekanizması oluşturma ve bir veri değişim stratejisi tasarlama adımlarını içerir, burada seçilen veri protokolü standardı bir veri etkileşim protokolüdür.

Tekniğin bilinen durumunda yer alan US2021081585A1 sayılı dokümanda, bir sistemin olaya dayalı simülasyonu için bir yöntemden bahsedilmektedir. Bu dokümanda simülasyon, bir birinci hesaplama birimi ve en az bir ikinci hesaplama birimi içeren bir bilgisayar sistemi üzerinde gerçekleştirilir; ilk hesaplama biriminin bir simülasyon süresi vardır; ikinci bilgi işlem birimi, bir işletim sistemi katmanına ve bir uygulama katmanına sahiptir; ikinci hesaplama birimi, işletim sistemi katmanında bir sistem saatine sahiptir; en azından ikinci bilgisayar birimi bir simülatör uygulamasını yürütür; simülasyon uygulamasında en az bir simülasyon nesnesi yürütülebilir; ve birinci bilgi işlem birimi bir olay kuyruğunu yönetir; olay kuyruğunda simülasyon adımı başına en az bir olay listelenir; ve olay, simülasyon nesnesi tarafından yürütülecek bir süreç ve sürecin yürütülmesi için sağlanan bir simülasyon süresi ile ilişkilidir; ikinci bilgi işlem biriminde sanal bir saat oluşturucu bulunur.

Teknikte yer alan uygulamalar incelendiğinde, uygulamalar arası senkronizasyon için kritiklik seviyelendirmesi yapılmadığı anlaşılmıştır. Bu kapsamda, simülatör projelerindeki uygulamaların eş zamanlı hareket etmesini sağlayan, dağıtık olarak çalışan simülatör uygulamalarını ağ üzerinden senkron eden, harici bir donanıma ihtiyaç duymayan, farklı frekanslarda çalışan simülatör uygulamalarının senkronizasyonunu sağlayabilen, yetişememe ve geri kalma problemleri için

uygulamalar arasında kritiklik seviyelendirmesi yapabilen ve bu sayede bir sonraki adım için hangi uygulamaların bekleneceği belirleyebilen bir yöntemin geliştirilmesi ihtiyacı doğmuştur.

Buluşun Amaçları

Bu buluşun amacı, simülatör projelerindeki uygulamaların eş zamanlı hareket etmesini sağlayan, dağıtık olarak çalışan simülatör uygulamalarını ağ üzerinden senkron eden, harici bir donanıma ihtiyaç duymayan, farklı frekanslarda çalışan simülatör uygulamalarının senkronizasyonunu sağlayabilen, yetişememe ve geri kalma problemleri için uygulamalar arasında kritiklik seviyelendirmesi yapabilen ve bu sayede bir sonraki adım için hangi uygulamaların bekleneceği belirleyebilen bir yöntemin gerçekleştirilmesidir.

Buluşun Ayrıntılı Açıklaması

Bu buluşun amacına ulaşmak için gerçekleştirilen senkronizasyon yöntemi ekli şekillerde gösterilmiştir.

Bu şekil;

Şekil 1: Yöntemin şematik gösterimidir.

Şekilde yer alan parçalar tek tek numaralandırılmış olup, bu numaraların karşılıkları aşağıda verilmiştir.

1. Saat
2. Uygulama
3. Eş zamanlama sunucu uygulaması
4. Eş zamanlama ajanı

Buluş konusu senkronizasyon yöntemi,

- Eş zamanlı sunucu uygulamasının (3), farklı frekansta çalışacak uygulamalar (2) için senkron frekanslarını belirlemesi,
- Belirlenen her senkron frekansı için ayrı yüksek öncelikli iş parçacıklarının oluşturulması,
- Her iş parçacığının işletim sistemi çağrıları ile aldığı yüksek çözünürlüklü süreölçer ile veya kendisi için atanan seri porttan sinyal geldiğinde UDP protokolü kullanarak belirlenen frekansta çok noktaya yayın yapması,
- Eş zamanlı sunucu uygulamasının (3); süre hassasiyeti için iş parçacıklarını uyutmama veya küçük uyku süreleri ile daha düşük işlemci kullanımı seçeneğinin olması,
- Yayınlanan mesajın içerisinde senkron frekansı ve döngü sayacı bulunması,
- Uygulamalar (2) içerisinde yer alan eş zamanlama ajanlarının (4) döngünün bitiminde bariyeri kilitlemesi ve sunucu uygulaması tarafından yayınlanan mesajın gelmesini beklemesi,
- Gelen mesajın içerisindeki frekans ve döngü sayacının kontrol edilmesi,
- Bu iki bilginin tutarlı olduğu durumda bariyerin açılması ve sonraki adıma geçilmesi,
- Eğer uygulama kritiklik seviyelendirmesi yapıldıysa, eş zamanlama sunucu uygulamasında (3) yer alan iş parçacıklarının bir sonraki mesajı yayınlamak için uygulamalardan " döngü adımını tamamladım" mesajını beklemesi,
- Kritik uygulamaların belirlenmesi,
- Eş zamanlama sunucu uygulamasının (3) gelen mesajın içerisindeki uygulama kimliğinin kritik bir uygulamaya ait olup olmadığını ve döngü sayacının ilgili iş parçacığından yayınlanan son senkron mesajı içerisindeki döngü sayacı ile eşitliğini kontrol etmesi,
- Senkron periyodu tamamlandığında kritik uygulamaların (2) hepsinden tamamladım mesajı alındığı zaman, uygulamaların içerisindeki eş zamanlama ajanları (4) için devam mesajı yayınlanması,

Adımlarını içermektedir.

Simülatör projelerinde; dağıtık mimaride çalışan uygulamalar senkron edilmediğinde beklenmedik sonuçlar ortaya çıkmaktadır. Bu durumun ortadan kalkması için uygulamaların aynı zaman dilimi içerisinde işlerini tamamlayıp sonraki adıma birlikte geçmesi beklenmektedir. Eş zamanlama sunucu uygulaması (3), ortamda çalışacak olan farklı frekanstaki uygulamalar için en uygun senkron frekans veya frekanslarını belirler.

Ağın, donanımın, işletim sistemi ve yükünün stabil olarak mümkün kıldığı bir maksimum çalışma frekansı vardır. Bu frekans, maksimum senkron frekansı olarak eş zamanlama sunucu uygulaması (3) konfigürasyonundan ayarlanır. (Varsayılan değer 300). Sonrasında eş zamanlama sunucu uygulaması (3), ortamda bulunan uygulama (2) frekansları argüman olarak verilerek çalıştırılır. Eş zamanlama sunucu uygulaması (3), bu frekansları en küçük ortak katı maksimum senkron frekansını geçmeyecek şekilde minimum sayıda gruplar. (Her grup bir senkron frekansıdır). Bu gruplardan toplamları en düşük olan seçilir. Buradaki amaç; maksimum senkron frekansını geçmeyen en az sayıda minimum senkron frekansı/frekanslarını belirlemektir.

Örn; 25, 50, 100, 60 için senkron frekansı 300 olur. (Ortak katların en küçüğü, OKEK)

25, 50, 100, 60, 120 için senkron frekansları 100 ve 120 olur. (OKEK 360 olacağından, 300'ün geçilmemesi gerekmektedir)

40, 50, 60 için senkron frekansları 120 ve 50 olur.([300,60], [120,50], [300,40] arasından (120,50) senkron frekansları olarak seçilir.)

Belirlenen her senkron frekansı için ayrı yüksek öncelikli iş parçacıkları oluşturulur.

Eş zamanlama sunucu uygulaması (3), belirlenen her senkron frekansı için; işletim sistemi çağrıları ile iş parçacıkları oluşturur. Süre ölçme ve UDP paket gönderme

işini bu iş parçacıkları üzerinde yapar. İş parçacığının önceliklendirilmesi ise eş zamanlama sunucu uygulamasının (3) Linux üzerinde çalıştığı durumda geçeli olup süre ölçüm hassasiyeti için eklenmiştirBu iş parçacığı önceliklendirmesi yine işletim sistemi çağrısı ile gerçekleştirilir.

Her iş parçacığı işletim sistemi çağrıları ile aldığı yüksek çözünürlüklü saat (1) (süreölçer) ile veya kendisi için atanan seri porttan sinyal geldiğinde UDP protokolü kullanarak belirlenen frekansta çok noktaya yayın yapar.

Burada tercihe bağlı iki farklı kullanım durumu vardır. İlki bilgisayarın işlemci saatini (1) referans alarak çalışan işletim sisteminin bize sağladığı yüksek çözünürlüklü saat (1) veya süreölçer olabilir. İkincisi ise eş zamanlama sunucu uygulamasının (3) çalıştığı bilgisayara haricen bağlanan ve istenilen frekansta seri porttan sinyal gönderen saat (1) olabilir. İlk durum eş zamanlama sunucu uygulaması (3) için varsayılan kullanımdır ve ekstra bir donanıma ihtiyaç duymaz. Bu işlem yine program tarafından yapılan işletim sistemi çağrıları gerçekleştirilir. İkinci durum ise isteğe bağlı olarak hassasiyeti arttırmak için kullanılabilir. Bu durumda eş zamanlama sunucu uygulaması (3) ilgili sürenin (senkron periyodu) geçip geçmediğini bu donanımın seri porttan yolladığı sinyal ile anlar. Çok noktaya yayın yapma (Multicast) kısmı ise; UDP(Datagram) protokolüne uygun açılmış soket vasıtası ile yapılır. (Bu işlem de aynı şekilde işletim sistemi çağrıları ile gerçeklenmektedir.)

Yayın yapan bu iş parçacıkları, işletim sistemi zamanlayıcısının bekleme kuyruğuna girmemesi için uyutulmaz.

Bu işleyiş de eş zamanlama sunucu uygulaması (3) konfigürasyonundan isteğe bağlı olarak değiştirilebilir. Varsayılan halinde; süre ölçüm hassasiyetinin maksimum olabilmesi için bu iş parçacıkları kilitlenmiyor/uyutulmuyor.(Spin-Lock, Busy-Wait) Yani işletim sistemi planlayıcısı (OS Scheduler) bu iş parçacıklarını uyku kuyruğuna almadan olabildiğince sıra vermeye çalışıyor. Bu

sayede iş parçacıkları işlemciden maksimum zamanı almış olur. Eş zamanlama sunucu uygulaması (3) konfigürasyonundan belirlenen ikinci kullanımda ise; bu periyod bekleme süresi boyunca ilgili iş parçacığı; daha deterministik uyku süresi belirlemek için 3 milisaniye tekrarlarla uyutulur. Bunun sonucunda eş zamanlama sunucu uygulamasının (3) işlemci kullanımı ve süre ölçer hassasiyeti düşer.

Eş zamanlama sunucu uygulaması (3) tarafından yayınlanan mesajın içerisinde senkron frekansı ve döngü sayacı bulunur. Uygulamalar (2) içerisinde yer alan eş zamanlama ajanları (4) döngünün bitiminde bariyeri kilitler ve eş zamanlama sunucu uygulaması(3) tarafından yayınlanan mesajın gelmesini bekler.

Çok parçacıklı çalışan simülatör uygulamalarında belli bir akış vardır. Bazı işlemler paralel bazı işlemler ise seri gerçekleştirilir. Buluş konusu yöntemde, eş zamanlama ajanı (4), bu döngünün sonunda çalışacak şekilde ayarlanmıştır. Yani eş zamanlama ajanının (4) işi bittiğinde bir sonraki döngü başlamış olmaktadır. Dolayısıyla her döngünün sonunda bir sonraki döngüye geçmek için ilgili mesaj beklenmektedir. Bariyer; çok iş parçacıklı uygulamalarda herhangi bir parçacığı çalışırken diğer iş parçacıklarını bekletmek/uyutmak için kullanılmaktadır. Kilitleme (Acquire Lock) / Kilit açma (Release Lock) gibi işlemler işletim sistemi çağırıları ile gerçekleştirilir. Bu durumda eş zamanlama ajanı (4) çalışırken uygulama (2) içindeki diğer simülasyon işlerini yapan iş parçacıkları bekler. Eş zamanlama ajanı (4) işi (mesaj bekleyip kontrol etme) bitiminde bu kilidi açar.

Eş zamanlama sunucu uygulamasından (3) gelen mesajın içerisindeki frekans ve döngü sayacı kontrol edilir. Bu iki bilginin tutarlı olduğu durumda bariyer açılır ve sonraki adıma geçilir.

Eş zamanlama sunucu uygulaması (3), senkron mesajını multicast olarak yayınlar. Dolayısıyla ortamdaki her eş zamanlama ajanı (4) içeren uygulama (2) bu mesajları alır. Bu yüzden mesaj geldiğinde uygulama ajanları alttaki iki kontrolü gerçekleştirir. Dolayısıyla eş zamanlama ajanı (4) mesaj geldiğinde;

1) Gelen mesaj içerisindeki senkron frekansı bilgisi benim senkron edileceğim frekans mı?

2) Gelen bu mesajın içerisindeki döngü sayacı tutarlı mı? (UDP olarak gönderilen mesajlarda herhangi bir sıra bozukluğu olma ihtimaline karşın kendi kaçıncı adımda olduğu bilgisi ile bu değeri kıyaslıyor. Eğer ki mesajın içindeki değer kendi adım sayacından küçükse dikkate almıyor)

Multicast kullanıp, her uygulama için ayrı ayrı göndermemekteki amaç; eş zamanlama sunucu uygulamasının (3) ağdaki yükü minimum tutmaktır.

Eğer uygulama kritiklik belirlenmesi yapıldıysa, eş zamanlama sunucu uygulamasında (3) yer alan iş parçacıkları bir sonraki mesajı yayınlamak için uygulamalardan " döngü adımını tamamladım" mesajını bekler. Eş zamanlama sunucu uygulaması (3), gelen mesajın içerisindeki uygulama kimliğinin kritik bir uygulamaya ait olup olmadığını ve yine bu mesajın içerisinde yer alan döngü sayacının ilgili iş parçacığından yayınlanan son senkron mesajı içerisindeki döngü sayacı ile eşitliğini kontrol eder.

Kritik belirlenmesi, eş zamanlama sunucu uygulamasını (3) kullanacak kişiler tarafından yapılır. Ortamdaki kritik olarak belirlenen birimler ilgili adımı bitirmeden diğer uygulamaların sonraki adıma geçmesini önlemek amaçlanmıştır. Örn; Ortamda A, B ve C sistemleri var. B sistemi kritik ve periyodunda işinin tamamlayamadı geç kaldı. Bu durumda eş zamanlama sunucu uygulaması (3) yeni adımın senkron mesajını göndermeyecek. Dolayısıyla A ve C B'yi beklemiş olacak. Herhangi bir sistem kritik olarak belirlenmediyse (Eş zamanlama sunucu uygulaması (3) konfigürasyonundan) eş zamanlama sunucu uygulaması (3) sadece zaman bağlı olarak ilerler ve sistemlerden "tamamladım" mesajını beklemez. Burada, kritik uygulamalardan, sunucu uygulaması için yayınlanan "tamamladım" mesajı uygulama kimliği ve döngü sayacından oluşur.

ŞEKİL - 1