Yazılım Özelliklerinin Enerji Tüketimi Üzerine Etkileri

Sedef Akınlı Koçak[1], Gülfem Işıklar Alptekin[2], Ayşe Başar Bener[3], Andriy Miranskyy[4], Emre Doğan[2]

[1] Ryerson University, Environmental Applied Science and Management, Canada
[2] Galatasaray Üniversitesi, Bilgisayar Mühendisliği Bölümü, İstanbul
[3] Ryerson University, Mechanical and Industrial Engineering, Canada
[4] IBM Toronto Lab, Canada

sedef.akinlikocak@ryerson.ca,gisiklar@gsu.edu.tr,
ayse.bener@ryerson.ca,andriy@ca.ibm.com,edogan@gsu.edu.tr

Özet. Yeşil yazılıma olan ilgi, hem akademik hem de endüstriyel dünyada gittikçe artmaktadır. Yeşil bir yazılım tasarlamak ve kodlamak, yazılım geliştirme şirketlerinin kurumsal sorumlulukları arasındaki yerini almıştır. Yazılım müşterileri, özellikle kurumsal müşteriler, seçimlerinde bu konuyu gözetmeye başlamışlardır. Çevresel sürdürülebilirliği olan bir yazılım geliştirmek meşakkatli bir süreçtir. Bu çalışmada, piyasada en bilinen veri tabanı yönetim sistemlerinden biri olan DB2 yazılımı ele alınmıştır. Deneysel çalışmalar için, DB2 üzerinde çalışan üç araç seçilmiştir. Bu araçlar aynı iş miktarı üzerinde kullanarak, altı farklı senaryo yaratılmıştır. Senaryoların birbirleriyle kıyaslanması için, üç grup yeşil ölçüt/kriter seçilmiştir: IT kaynak ölçütleri, yaşam döngüsü ölçütleri ve sistem enerji kullanımı ölçütleri. Sonuçlar, bu araçların yazılım sisteminin enerji etkinliği üzerine farklı bireysel ve bileşik etkileri olduğunu göstermiştir. Elde edilen sonuçların, yazılım sistemi performansı ve yazılım enerji tüketimi arasındaki ödünleşim modellerinde kullanılması mümkün olabilir.

Anahtar Sözcükler Yeşil Bilişim, Yeşil IT, Yeşil Yazılım, Enerji Tüketimi, Çevresel Sürdürülebilirlik, Enerji Etkinliği

1 Giriş

Yeşil bilişim/IT, enerjiyi daha etkin kullanan donanımlar önerdiğinden, çevresel sürdürülebilirliğe katkısı olan bir çalışma alanıdır. Bu mantıkla bakıldığında, yeşil bilişime, işlevselliğini kaybettirmeden daha az enerji harcayan yazılım tasarlayıp üretmek de dahil edilebilir. Kern vd., yaptıkları çalışmada, sadece yazılımın da, genel enerji tüketimine azımsanmayacak derecede etki ettiğini göstermiştir [1]. Son yıllarda, yazılım şirketleri karbon ayak izlerini küçültmeye çalışarak, yüksek rekabette bir adım öne geçmeye çalışmaktadırlar. Bir yazılımın yeşilliği, o yazılımın geliştirilmesi, teslimi ve bakımı süreçlerinin tümünü birden kapsamalıdır [2]. Global karbon

salınımını doğrudan etkilediği için, enerjinin etkin kullanımı, yeşil yazılımın da en önemli konusudur.

Daha önceki çalışmamızda, bir yazılım ürününün kullanıcı gereksinimlerine göre modernleştirme sürecindeki enerji tüketimi incelenmiştir [3]. Enerji tüketimine doğrudan etkisi olduğunu düşünmemiz sebebiyle, veri sıkıştırma aracı ele alınmıştır. Yazılımın işlevselliğini arttırmak ve enerji tüketimini azaltmak arasındaki ödünleşim üzerinde durulmuştur. Bu çalışmada ise gerçek hayat senaryolarına daha da yakınlaşmaya çalışılıp, araştırma sorusu olarak şu belirlenmiştir: "Yazılım araçlarının enerji tüketimi üzerine bireysel ve bileşik etkileri nelerdir?" Bu soruya cevap verebilmek ve yazılımın tükettiği enerji miktarını somut olarak ölçebilmek için yeşil ölçütler/kriterlerden faydalanılmıştır. Dolayısıyla, bu çalışmadaki ikinci araştırma sorusu da şu olmuştur: "Bir yazılım sisteminin enerji etkinliğini değerlendirmek için faydalanılabilecek yeşil ölçütler nelerdir?"

Ölçümleri gerçekleştirmek için, çok kullanılan veri tabanı yazılım sistemlerinden olan DB2 seçilmiştir. Deneylere, ilk çalışmamıza ek olarak iki yazılım aracı daha eklenmiştir. Bu sayede, daha ayrıntılı deney senaryoları yaratabilip; yazılım araçlarının bireysel ve bileşik etkilerini daha iyi görebilmek mümkün olmuştur. Birbirinden farklı altı deney senaryosu yaratılıp, her bir senaryodaki CPU kullanım oranı, I/O bekleme oranı, iş performansı ve toplam enerji tüketimi üstüne yoğunlaşılmıştır. Her bir senaryonun farklı enerji tüketim ve kaynak kullanım profili ürettiği saptanmıştır. Bu bilgiler, yöneticilerin yazılım geliştirme süreçlerini daha yeşil kılabilmek için kurmak zorunda oldukları ödünleşim modellerinde etkin olarak kullanılabilecektir. Kullanılmasını önerdiğimiz yeşil ölçütler, yazılım geliştiricilerin, ürettikleri yazılımın ne kadar yeşil olduğunu somut olarak ölçmelerini sağlayacaktır.

Makalenin 2. bölümünde, konuyla ilgili yapılan çalışmalar özetlenmiştir. 3. bölümde deneylerde kullanılan yeşil ölçütler, 4. bölümde ise üzerinde çalışılan deney altyapısı anlatılmıştır. 5. bölümde elde edilen sonuçlar verildikten sonra, bu sonuçlar tartışılarak makale sonlandırılmıştır.

2 İlgili Akademik Yazın

Enerji yönetimi teknikleri, bilgisayar sistemlerinin birçok seviyesinde kullanılmaktadır. Enerji yönetimi konusundaki basit yaklaşım, kullanılmayan bileşeni düşük enerji moduna geçirmek veya tamamen pasif hale getirmektir. Düşük enerji harcayan yazılım konusundaki öncü çalışmalardan ikisinde, enerji komut seviyesinde incelenmiş ve her bir komutun birim enerji harcadığı fikri ortaya atılmıştır [4-5]. Yeşil ölçütler, enerji tüketimini değişik sistem seviyelerinde ölçmek için kullanılmıştır [6]. Akademik yazındaki çoğu araştırma, donanım ve yazılımın güce olan etkilerini ölçmeyi amaçlamıştır. Yakın zamanda yayınlanan çalışmalardan birinde yazarlar, yazılım alanında çevresel sürdürülebilirliği hedefleyen modeller öneren çalışmaların eksikliğini belirtmişlerdir [7]. Geliştirdikleri 'GreenSoft' adlı modelle, yeşil yazılım ölçütleri önermiş ve yeşil yazılım tasarımı ve geliştirme süreci ile ilgili teorik bir model ortaya koymuşlardır. Ancak bu zaman kadar, yeşil ölçütleri kullanarak bir yazılımın enerji etkinliğini ölçen bir çalışmaya rastlanılmamıştır.

3 Kullanılan Yeşil Ölçütler

Yeşil ölçüt olarak önerilen birçok ölçüt olsa da [8], bu çalışma için sistemi bir bütün olarak kapsayan temel bir ölçüt kümesi seçilmiştir. Sistemin enerji etkinliğini ölçmek için kullanılan yeşil ölçütler Tablo 1'de özetlenmiştir.

Tablo 1. Yeşil ölçütler

Seçilen Yeşil Ölçütler	Birim
IT Kaynak Kullanım	
CPU kullanım oranı	%
I/O kullanımı	%
Saklama birimi kullanımı	%
Yaşam Döngüsü	
Uygulama performansı (QoSL)	Tamamlanan faydalı iş sayısı / kWh
Enerji Etkisi	
Sistem enerji tüketimi	kWh
Uygulama enerji etkinliği	W/tamamlanan komut sayısı

Bazı ölçütlerin hesaplanış biçimleri şu şekildedir:

Uygulama performansı = Tamamlanan faydalı iş / Enerji tüketimi (1)
Uygulama enerji etkinliği = Enerji tüketimi / Tamamlanan komut sayısı (2)
Saklama birimi kullanımı = Kullanılan disk alanı / Ayrılan disk alanı (3)
Alan kazancı = 1 – (Sıkıştırılan veri boyutu / Sıkıştırılmamış veri boyutu) (4)

Bu ölçütlerin sistemin genel performansı ve enerji etkinliği hakkında fikir verebileceğine inanılmıştır. Her bir senaryoya ait performans ise, uygulama performansı ölçütü ile hesaplanmaktadır.

4 Vaka Analizi

Deneylerin gerçekleştirildiği, özellikle modern veri tabanı sistemlerinde kullanılan üç farklı aracın kombinasyonlarının oluşturduğu altı farklı senaryo Tablo 2'de özetlenmiştir. Yazılım olarak, Linux, Unix ve Windows 10.1 platformları üzerinde çalışan DB2 seçilmiştir. DB2, 1992 yılından beri piyasada yer alan, kapsamlı bir veri tabanı yönetim sistemidir.

4.1 Kullanılan Yazılım Araçları

Yazılım araçlarını kısaca özetleyelim:

a) **DB2 Adaptive Data Compression (*C*):** Saklama alanları çoğu zaman veri tabanlarının en çok maliyet yaratan bileşeni olmaktadır. Saklama alanında yaratılacak

ufak bir iyileştirme, tüm sistemde büyük bir maliyet azalmasına yol açabilmektedir. DB2'nun son sürümünde, gelişmiş bir veri sıkıştırma eklentisi (*adaptive data compression*) bulunmaktadır [9].

b) **DB2 Design Advisor: Indexes (*I*):** 'Design advisor' olarak isimlendirilen araç, otomatik olarak sorguları analiz edip, yapılarına bakarak gerekli nesneleri (örneğin tablo indisleri) önerir. Benzer bir analizi, veri tabanı yöneticisi de yapabilir; ancak sorgu sayısı ve karmaşıklığı arttıkça iş yükü çok ciddi artmaktadır. '*Index advisor*', bu aracın içindeki alt araçlardan biridir ve veri tabanından veri çekme hızını arttıran bir veri yapısıdır. Bu araç genelde, fazladan saklama alanı ve işlemci gücü (CPU) gerektirmektedir.

c) **DB2 Design Advisor: Indexes and Materialized Query Tables (*I+MQT*):** 'Design advisor' aracının alt araçlarından bir diğeridir. Çok işlem gerektiren bazı sorgu işlemlerinin (ör: *join*, *sort*) tekrar tekrar yapılmasını engelleyerek, sorgu performansını üssel seviyelerde iyileştirebilir. Bu araç da fazladan saklama alanına ve işlemci gücüne gereksinim duyar.

4.2 Deney Ortamı

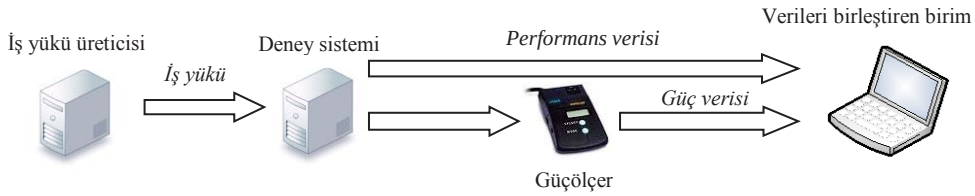
Analizleri gerçeleştirmek için TPC-H (*Transaction Processing Council*)'dan alınan OLAP (*online analytical processing*) tipinde bir iş yükü kullanılmıştır. OLAP sorguları kullanmanın avantajı, daha karmaşık ve daha büyük boyutlu verileri, daha hızlı bir şekilde işlemeye olanak tanıması olmuştur. TPC, veri tabanı performansını ölçmek için endüstrinin kullandığı bir standarttır. Veri tabanı içinde sekiz bağımsız tablo içinde, 1 GB veri yer almaktadır. İşle ilgili, oldukça karmaşık, toplam 240 sorgudan oluşmaktadır. Deney kapsamında, sorgular iki saat boyunca sırayla ve aralıksız şekilde, Linux Ubuntu v.12.04 işletim sistemine sahip, 4 GB RAM'i ve çift çekirdekli işlemcisi olan bir bilgisayar üzerinde çalıştırılmıştır. Her bir senaryo için, aynı sorgular dört kere çalıştırılmış, son üç seferde hemen hemen aynı sonuçlar elde edildiği saptanmış ve son üç seferin ortalamaları alınmıştır.

4.3 Enerji Tüketimi Ölçme Yöntemi

Bir yazılımın enerji tüketimini ne zaman ve nasıl ölçülmesi gerektiğini belirlemek, oldukça meşakkatli bir iştir. Akademik yazında farklı biçimler yer alsa da, bu çalışma için kullanılan test ortamı, Şekil 1'de verilmiştir. İş yükü, iş yükü üreticisi tarafından üretilir ve deney sistemine yollanır. Deneyler sistemin üzerinde doğrudan çalıştırılır. Güçölçerin okuduğu değerler ve sisteme ait tüm veriler (enerji tüketimi verileri, I/O sayıları, saklama birimi kullanım oranları, CPU kullanımı verileri), değerlendirilmek üzere veri toplayan birime gönderilir.

Tablo 2. Yazılım araçlarının aktif olup olmamasına göre değişen senaryolar

	"Design Advisor (DA) Objects" yok	**"Indexes suggested by DA" var**	**"Indexes & MQTs suggested by DA" var**
Veri sıkıştırma yok	C- DA- (Senaryo 1)	C- I+ (Senaryo 2)	C- I+ MQT (Senaryo 3)
Veri sıkıştırma var	C+ DA- (Senaryo 4)	C+ I+ (Senaryo 5)	C+ I+ MQT (Senaryo 6)



Şekil 1. Deney ortamı

5 Analiz Sonuçları

Önceki çalışmamızda [3], sadece veri sıkıştırması aracının sistem enerji kullanımı üzerindeki etkileri izlenmiştir. Elde edilen sonuçlara göre, veri sıkıştırması aracını kullanarak, saklama alanı kullanımı %61 oranında, birim iş için tüketilen enerji ise %34 oranında azalmıştı. Ayrıca veri sıkıştırıldığında, saatte işlenen komut sayısında %97'lik bir artış olmuştu. Bu çalışmada ise, gerçek hayat senaryolarına daha çok benzemek için, yazılıma iki araç daha eklenmiştir. Tablo 3, sistem boşken elde edilen ölçümleri, Tablo 4 ise tüm deney sonuçlarını içermektedir.

Tablo 3. Sistemin boşken ölçüm sonuçları

	Toplam CPU kullanım oranı (%)	**Toplam I/O bekleme oranı (%)**	**Sistem enerji kullanımı (kWh)**
Boş (IDLE)	0.09	0.09	0.14

5.1 IT Kaynak Kullanımı Ölçütleri Sonuçları

Kaynak kullanımı deyince akla ilk CPU kullanımı gelse de, enerji tüketimini sadece CPU kullanımı olarak görmemek gerekir. I/O sırasında beklemeler ve saklama alanı kullanımı da kaynak kullanımları arasındadır. Şekil 2(a), kullanıcı sayısı ne olursa olsun, veri sıkıştırma aracının CPU kullanımını çok arttırdığını göstermektedir. (I) ve (MQT) araçlarının bir arada kullanımının CPU kullanımını azalttığı saptanmıştır (Senaryo 1 vs. Senaryo 3). Tüm araçların bir arada kullanıldığı senaryoda, CPU kullanımının ciddi derecede arttığı gözlemlenmiştir (Senaryo 6 vs. Senaryo 1). Bu deneydeki ilginç bulgu ise, (C) ve (I+MQT) araçlarının bir arada kullanıldığı durumdur (Senaryo 3 vs. Senaryo 6). Kullanıcı sayısı değişiminin, CPU kullanım oranını da

değiştirdiği izlenmiştir. Örneğin, tek kullanıcılı sisteme veri sıkıştırma aracı (C) eklendiğinde, CPU kullanımı dört kat artmıştır. Ancak, iki, dört ve sekiz kullanıcılı sistemdeki CPU kullanımı sırasıyla 1.8, 1.1 ve 4 kat artmıştır (Senaryo 1 vs. Senaryo 4). Şekil 2(b)'deki toplam I/O bekleme oranı, I/O işlemleri sırasında CPU'nun beklemek zorunda kaldığı döngü oranını göstermektedir Örneğin, Senaryo 1'deki tek kullanıcılı durumda, döngülerin %45.4'lük kısmında CPU I/O işlemlerinin tamamlanmasını beklemektedir (Tablo 4). Veri sıkıştırma aracı sayesinde, sabit diskten yapılması gereken okuma isteği miktarı azaldığı için, I/O bekleme oranı tek kullanıcılı sistem için %28, dört kullanıcılı sistem için %95 ve 8 kullanıcılı sistem için ise %98 azalmıştır (Tablo 4). Sonuçlar, (I) ve (MQT) araçlarının veri sıkıştırması (C) ile birlikte kullanıldığında I/O beklemeleri üzerinde olumlu iyileşme sağladığını göstermektedir (Senaryo 3 vs. Senaryo 6). Yüksek I/O bekleme oranı, işlenen verinin tamamının belleğe alınamaması demektir. Belleğe getirilemeyen veri, yüksek miktarda okuma ve yazma isteğine sebep olmaktadır. Bu istekler sabit diskten gerçekleştirildiği için, beklemeler olmaktadır (Senaryo 1, 2 ve 3). Kalan üç senaryodaki istisnai durum, belleğe getirilmesi gereken verinin ciddi miktarda azalması ile açıklanabilir. Daha fazla kullanıcı, daha fazla okuma isteği demek olduğundan, CPU'nun daha fazla çalışması, dolayısıyla daha az beklemesi demektir. Deneylerde, sadece Senaryo 4 dışında, her durumda daha fazla saklama birimine ihtiyaç duyulduğunu gördük (Şekil 3(a)). Sonuçlar bize, CPU ve saklama birimi kullanımı arasında bir ödünleşim olduğunu gösterdi. Senaryo 3'e bakıldığında, toplam CPU kullanım oranının en düşük seviyede olduğu görülmektedir; ancak aynı senaryonun bellek kullanımı en üst seviyededir. İş yükünün artışı, birim zamanda işlenen ortalama komut sayısını arttırmış; bu da CPU kullanım oranını arttırmıştır (Şekil 2(a)).

Tablo 4. Senaryolara ait ölçümler

	Saklama (%)	Komut sayısı				Toplam CPU kullanım oranı (%)				Toplam I/O bekleme (%)				Uygulama performansı (#faydalı iş/kWh)				Uygulama enerji etkinliği (W/işlem sayısı)				Sistem enerji kullanımı (kWh)			
Kullanıcı		**1**	**2**	**4**	**8**	**1**	**2**	**4**	**8**	**1**	**2**	**4**	**8**	**1**	**2**	**4**	**8**	**1**	**2**	**4**	**8**	**1**	**2**	**4**	**8**
C- DA-	0	1118	8079	11358	7319	8.5	54.3	83.9	32.7	45.4	27.6	10.2	53.5	6559	35832	43687	38519	0.15	0.03	0.02	0.02	0.17	0.23	0.26	0.19
C- I+	1.76	2598	3445	3907	2496	12.3	18.3	21.7	15.7	42.7	49.3	51.3	64.3	15755	20264	22985	15124	0.06	0.05	0.04	0.07	0.17	0.17	0.17	0.17
C- I+ MQT	2.94	3389	4519	3678	4121	6.6	8.9	8.9	9.7	45.3	54.7	58.8	63.2	21184	29207	23726	26587	0.05	0.03	0.04	0.04	0.16	0.16	0.16	0.15
C+ DA-	-0.49	3846	12641	12264	11521	32.2	99.4	98.7	98.2	32.5	0	0.5	1	19230	41138	41573	39054	0.05	0.02	0.03	0.03	0.2	0.3	0.29	0.29
C+ I+	0.48	6527	15582	14804	14842	33.2	83.9	85.1	98.1	19.4	11.1	11.3	1.3	31859	57713	54830	53007	0.03	0.02	0.02	0.02	0.21	0.27	0.27	0.28
C+ I+ MQT	1.35	25648	48704	46779	44888	49.8	99.2	97.7	96.7	0.5	0.2	1.5	2.4	100568	162348	158575	154786	0.01	0.01	0.01	0.01	0.26	0.3	0.29	0.29

5.2 Yaşam Döngüsü Ölçütleri Sonuçları

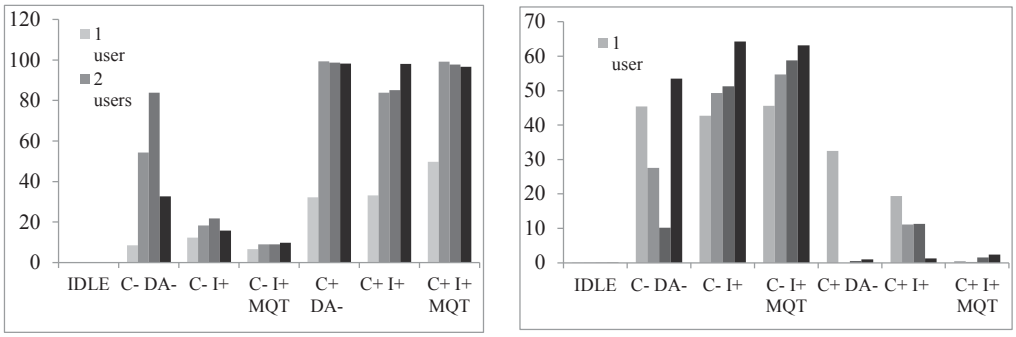
Uygulama performansı, enerji tüketimi başına gerçekleştirilen faydalı iş olarak tanımlanmıştır. Son üç senaryodaki (özellikle Senaryo 6) uygulama performansı artışı, ortalama CPU kullanımı artışından ve birim zamanda işlenen ortalama komut sayısı artışından kaynaklanmaktadır. Bu da göstermektedir ki, birim zamanda işlenen komut sayısındaki artış, CPU kullanım oranındaki artıştan çok daha yüksektir. Teorik olarak, iş yükü miktarı sabitlendiğinde, donanım sadece bu iş yüküyle ilgilenir ve genelde CPU kullanım oranı görmezden gelinir. Şekil 2(a), Senaryo 6'da görülebileceği gibi, tüm araçları kullanmak faydalı gibi gözükmese de, bunun böyle olmadığını Şekil 3(b), Senaryo 6'da görüyoruz. Uygulama performansı en yüksek değerine, tüm araçlar bir arada kullanıldığı zaman ulaşmaktadır. Çünkü aslında, donanım birçok iş arasında paylaştırılmaktadır (özellikle sanallaştırmanın olduğu, bulut bilişim altyapılarında).

5.3 Enerji Ölçütleri Sonuçları

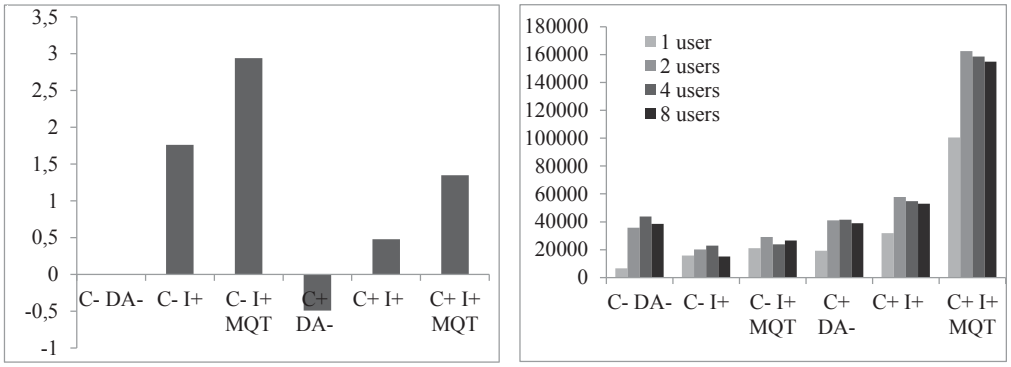
Uygulama enerji etkinliği, bir tek komutu işlemek için gereken enerji anlamına gelmektedir. Bu ölçüt sayesinde, bir işi tamamlamak için tüketilmesi gereken enerji miktarı ortaya konulabilir. Birim zamanda tamamlanan ortalama komut sayısı arttıkça, komut başına tüketilen enerji miktarında bir azalma görülür (Tablo 4). Şekil 4(a), her senaryoya ait enerji etkinliği sonuçlarını özetlemektedir. Uygulama performansı sonuçlarına benzer şekilde, üç aracın birlikte çalıştırılmasının bileşik etkisi çok barizdir (Senaryo 1 vs. Senaryo 6, özellikle tek kullanıcılı durumda). Bunun yanında, kullanıcı sayısı arttıkça, faydalı iş miktarının da arttığı daha önce söylenmişti. Bunun göze çarpan faydası, tüm araçların pasif konuma getirildiği Senaryo 1'de gözükmektedir. Enerji bakımından en dikkat çekici olan, tüm araçların etkinleştirildiği Senaryo 6'dır. Şekil 4(b), her senaryoda yazılım sisteminin toplamda ne kadar enerji harcadığını kWh cinsinden göstermektedir. Açıkça görülmektedir ki, (I) ve (MQT) araçları bir arada çalıştırıldığında enerji kazançları olmaktadır (Senaryo 1 vs. Senaryo 3).

6 Sonuç ve Öneriler

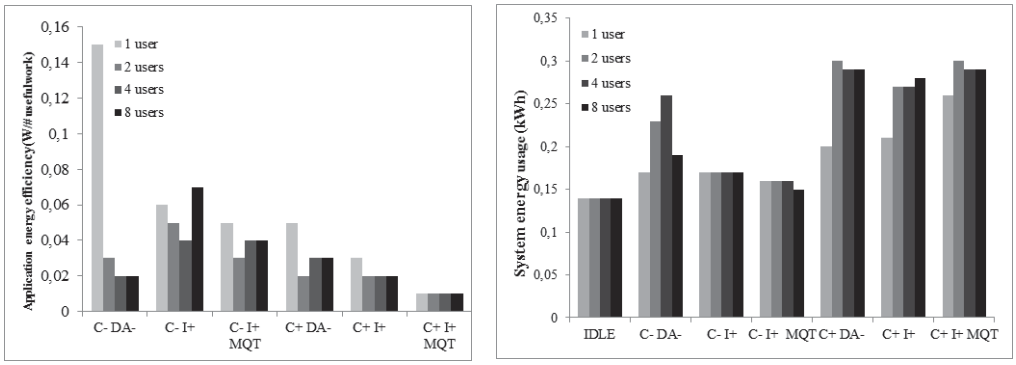
IT şirketleri, hem kendi maliyetlerini kısmak, hem de çevresel sürdürülebilirliğe katkıda bulunmak amacıyla, yeşil/sürdürülebilir stratejilere gittikçe daha fazla ilgi duymaya başladılar. Yakın geçmişte yapılan çalışmalar, sürdürülebilirliğe katkının sadece donanımla değil, aynı zamanda yazılımla da yapılabileceğini göstermektedir. Yazılım geliştirme süreci, müşteri talepleri ve enerji gereksinimleri arasında yapılması gereken bir ödünleşim gerektirir olmuştur.



Şekil. 2. (a) Toplam CPU kullanım oranı (%) (b) Toplam I/O bekleme oranı (%)



Şekil. 3. (a) Saklama birimi kullanımı(%) (b) Uygulama performansı (işlem sayısı/kWh)



Şekil 4. (a) Uygulama enerji etkinliği (W/faydalı iş sayısı) (b) Uygulama enerji performansı

Bu makalede, yazılım araçlarının, yazılım enerji etkinliği üzerine bireysel ve bileşik etkileri gösterilmiştir. Sistemin yeşil performansı, kullanılan farklı ölçütlerle belirlenmiştir. DB2'ye ait MQT aracının veriyi bir yerde tutup oradan türetebileceği yerde, gereksiz yere başka yerlerde de tuttuğunu ve bunun da fazladan saklama alanı israfı yarattığı görülmüştür. Veriyi diskten belleğe atmak, veri tabanının en yavaş yaptığı

işlemlerinden biridir. Diskte sıkıştırılmış veri tutmak, sıkıştırılmamış birim veri açısından düşünüldüğünde daha az I/O işlemi yapılmasını gerektirir. Dolayısıyla, yoğun I/O gerektiren işlerde, sorgu işleme zamanlarında gözle görülür bir iyileşme gözlemlenebilir. Ayrıca, DB2 prosesleri veriyi bellekte sıkıştırılmış şekilde depolayabilmektedir. Bu da, sıkıştırılmamış şekilde tutmaya kıyasla daha az bellek gereksinimi demektir. Böylece, fiziksel bir ek belleğe ihtiyaç duymadan, veri tabanı belleğini arttırmak veya arta kalan belleği başka işlemler için kullanmak mümkündür. Bunun da, veri tabanı performansına doğrudan olumlu etki yapacağı açıktır. Veri sıkıştırma aracının (C) ve (I+MQT) aracının birlikte kullanıldığında ise, CPU kullanımının aşırı derecede arttırdığı gösterilmiştir. Bu durumda, işlenen birim iş yükü de arttığı için, uygulama performansı üzerinde de bir artış gözlemlenmiştir. Uygulama performansının da, sistem enerji etkinliği üzerine doğrudan etkisi vardır. Diğer taraftan bakıldığında, birim iş yükü artışı, daha fazla saklama alanı gerektirmektedir. Bu durum bir ödünleşim problemi olarak görülebilir. İleriki çalışmalarda, bu tarz çevresel sürdürülebilirlikle ilgili ödünleşim problemleri, yazılım geliştirme sürecine dahil edilebilir. Kurulan modelle, verilen bütçe, kaynak ve enerji kısıtları uyarınca, yeşil ölçütlerin optimum değerleri hesaplanabilir.

7 Destek

Bu araştırma, 13.401.004 numaralı Galatasaray Üniversitesi Bilimsel Araştırma Projesi tarafından finansal olarak desteklenmetedir.

8 Kaynaklar

1. Kern, E., Dick, M., Johann, T., Nauman, S., 2011. "Green Software and Green IT: An End Users Perspective", **Engineering: Information Technologies in Environmental Engineering New Trends and Challenges**, Springer Berlin Heidelberg, 198-211 (2011).
2. Taina, J., "How green is your software?" **Software Business**, 51:151-162 (2010).
3. Akınlı Koçak, S., Miransky, A., Işıklar Alptekin, G., Başar Bener, A., Cialini, E. "The impact of improving software functionality on environmental sustainability", **Proceedings of the First International Conference on ICT for Sustainability (ICT4S)**, Zurich, Switzerland, 104-109 (2013).
4. Tiwari, V., Malik, S., Wolfe, A. "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization", **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, 24:437–445 (1994).
5. Tiwari, V., Malik, S., Wolfe, A., Tien-Chien Lee, M."Instruction Level Power Analysis and Optimization of Software", **The Journal of VLSI Signal Processing**, 13:223–238 (1996).
6. Cappiello, C., Fugini, M., Pernici, B., Plebani, P. "Green Information systems for Sustainable IT" **Information Technology and Innovation Trends in Organizations, Physica-Verlag HD**, 153-160 (2011).

7. Naumann, S., Dick, M., Kern, E., Johann, T., "The GREENSOFT model: A reference model for green and sustainable software and its engineering", **Sustainable Computing: Informatics and System**, 1:294– 304 (2011).
8. Kothiyal, R., Tarasov, V., Sehgal, P., Zadok, E., "Energy and performance evaluation of lossless file data compression on server systems", **Proceedings of ACM SYSTOR 2009: The Israeli Experimental Systems Conference**, 4-16 (2009)..
9. Bhattacharjee, B., Lim, L., Malkemus, T., Mihaila, G., Ross, K., Lau, S., McArthur, C., Toth, Z., Sherkat, R.. "Efficient index compression in DB2 LUW", **Proceedings of the VLDB Endow**. 2(2):1462-1473 (2009).