T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ADA PROGRAMLAMA DİLİNDE GERÇEK ZAMANLI KIYASLAMA UYGULAMA KÜMESİ

YÜKSEK LİSANS TEZİ

Özge GÖKÇE

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Dr. Öğr. Üyesi Veysel Harun ŞAHİN**

Haziran 2019

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ADA PROGRAMLAMA DİLİNDE GERÇEK ZAMANLI KIYASLAMA UYGULAMA KÜMESİ

YÜKSEK LİSANS TEZİ

Özge GÖKÇE

Enstitü Anabilim Dalı **BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**

Bu tez 10 Haziran 2019 tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

Dr. Öğr. Üyesi	**Dr. Öğr. Üyesi**	**Dr. Öğr. Üyesi**
Veysel Harun ŞAHİN	**Abdullah Sevin**	**Bahadır Hiçdurmaz**
Jüri Başkanı	**Üye**	**Üye**

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Özge GÖKÇE

10.06.2019

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Dr. Öğr. Üyesi Veysel Harun ŞAHİN'e teşekkürlerimi sunarım.

Maddi ve manevi desteğiyle her zaman yanımda olan, vatana ve millete faydalı işler yapmam konusunda daima teşvik eden, çalışkan ve ahlaklı bir birey olarak yetiştiren sevgili babam ve anneme en içten teşekkürlerimi sunarım.

İÇİNDEKİLER

TEŞEKKÜR ..……..... i
İÇİNDEKİLER ……..... ii
SİMGELER VE KISALTMALAR LİSTESİ ……..... iv
ŞEKİLLER LİSTESİ ……..... v
TABLOLAR LİSTESİ …….. vi
ÖZET ……. vii
SUMMARY …….. viii

BÖLÜM 1.
GİRİŞ……..... 1

BÖLÜM 2.
WCET ……..…….....**.** 3

BÖLÜM 3.
KIYASLAMA UYGULAMALARI……... 5
3.1. Kıyaslama Uygulama Kümesi Kavramı…….....……. 5
3.2. Literatürde Yer Alan Kıyaslama Uygulama Kümeleri…….. 5

BÖLÜM 4.
ADA PROGRAMLAMA DİLİ…… 8

BÖLÜM 5.
ABENCH ……... 10
5.1. Ada Dilinde, Kıyaslama Uygulama Kümesi…… 10

5.1.1. Özellik matrisi ……. 11
5.2. Kıyaslama Uygulamalarının Detayları……. 13
5.2.1. Uygulamalar hakkında dizin yapılandırması……. 15

BÖLÜM 6.
TARTIŞMA……. 17

BÖLÜM 7.
SONUÇ…… 18

KAYNAKLAR ……. 19
ÖZGEÇMİŞ ……... 21

SİMGELER VE KISALTMALAR LİSTESİ

A	: Dizi (Array)
ABench	: Ada programlama dilinde kıyaslama uygulama kümesi
BLO	: Bit düzeyinde işlemler (Bit level operations)
D	: Karar (Decision)
DMA	: Dinamik hafıza (Dynamic memory allocation)
ER	: Harici kütüphane (External routine)
FPO	: Ondalıklı sayı işlemleri (Floating point operation)
IF	: Girdi dosyası (Input file)
IO	: Tam sayı işlemleri (Integer operation)
IVAL	: Girdi değeri (Input value)
IVEC	: Girdi vektörü (Input vector)
L	: Döngü (Loop)
MP	: Çok yollu (Multi path)
MT	: Çok iş parçacıklı (Multi threaded)
NL	: İç içe döngü (Nested loop)
R	: Özyinelemeli (Recursion)
SP	: Tek yollu (Single path)
ST	: Tek iş parçacıklı (Single threaded)
WCET	: En kötü durum yürütme süresi (Worst-case execution time)

ŞEKİLLER LİSTESİ

Şekil 5.1. TreeTraversal programı için çağrı grafiği..... 15
Şekil 5.2. TreeTraversal programı için kapsam hiyerarşi grafiği…… 16
Şekil 5.3. Dizin yapısı..... 16

TABLOLAR LİSTESİ

Tablo 3.1. Kıyaslama uygulama kümeleri özellikleri……. 7
Tablo 5.1. Program özellik açıklamaları……. 12
Tablo 5.2. Program özellik matrisi…….. 13

ÖZET

Anahtar kelimeler: Gerçek Zamanlı Sistemler, Ada, Kıyaslama, Yazılım Mühendisliği

Gerçek zamanlı sistemler teknolojinin de gelişmesiyle endüstri, otomotiv, haberleşme ve havacılık gibi pek çok alanda yaygın olarak kullanılmaktadır. Bilgi sistem teknolojilerinin gelişimiyle gerçek zamanlı sistemlerin kullanımı hayatın her alanında artmıştır ve bu sistemler geliştirilirken C dilinin yanında Ada, Java, Assembler programlama dilleri de kullanılmaktadır. Gerçek zamanlı sistemlerin kullanımının artması ile bu sistemlerin karşılaştırma ve değerlendirme ihtiyacı oluşmuştur.

En kötü yürütme süresi analizi (WCET), gerçek zamanlı sistem geliştirme sürecinin en önemli parçalarındandır. Bu analizi gerçekleştirebilmek için WCET araçları ve yöntemleri geliştirilmektedir. Yeni WCET araçları ve yöntemleri geliştiren araştırmacılar, geliştirdikleri sistemleri değerlendirmek ve diğer sistemlerle karşılaştırmak amacıyla kıyaslama uygulamalarına ihtiyaç duyarlar. Bu amaçla, oluşturulmuş olan kıyaslama uygulama kümelerini kullanarak bu değerlendirmeyi gereçekleştirirler. Mevcut kıyaslama uygulama kümelerinin çoğu C dili kullanılarak kodlanmıştır. Gerçek zamanlı sistemler geliştirilirken C, Ada, Java, Assembly dilleri yaygın olarak kullanılmaktadır. Dolayısıyla farklı programlama dillerinde yazılmış kıyaslama uygulamalarına da ihtiyaç bulunmaktadır.

Bu tezde WCET araçlarının ve yöntemlerinin değerlendirilmesine ve kıyaslanmasına yardımcı olmak amacıyla, Ada dilinde kodlanmış olan ABench isminde yeni bir kıyaslama kümesi geliştirilmiştir. Geliştirilen kıyaslama uygulama kümesi, Ada programlama dilinde yazılmış çeşitli kıyaslama uygulamaları içermektedir. Oluşturulan kıyaslama uygulamaları; sıralama, arama, olasılık ve bir çok farklı alanda temel algoritmanın uygulamalarını içerir.

REAL-TIME BENCHMARK APPLICATION COLLECTION IN ADA PROGRAMMING LANGUAGE

SUMMARY

Keywords: Real-Time Systems, Ada, Bechmark, Software Engineering

Real-time systems are widely used in many areas such as industry, automotive, communication and aviation with the development of technology. With the development of information system technologies, the use of real-time systems has increased in all areas of life. In developing real-time systems C, Ada, Java and Assembler programming languages are used. The use of real-time systems increases the need for comparison and evaluation of these systems.

The worst execution time analysis (WCET) is one of the most important parts of the real-time system development process. To perform this analysis, WCET tools and methods are being developed. Researchers who develop new WCET tools and methods need benchmarks in order to evaluate the systems they have developed and compare them with other systems. Most of the available benchmarks are coded using the C language. C, Ada, Java, Assembly languages are widely used when developing real-time systems. Therefore, benchmarking suites written in different programming languages are also needed.

In this thesis, a new set of comparisons, named ABench, has been developed in order to assist the evaluation and comparison of WCET tools and methods. The developed benchmark suite contains several benchmark applications written in the Ada programming language. It includes applications of sort, search, probability and basic algorithm in many different fields.

BÖLÜM 1. GİRİŞ

Kıyaslama uygulamaları, bilgisayar sistemlerini, gerçek zamanlı sistemleri, algoritmaları ve bunun yanında yazılım sistemi kullanılan pek çok alandaki çalışmaları kıyaslamak ve değerlendirmek için hem geliştiriciler hem de araştırmacılar tarafından kullanılmaktadır. Kıyaslama programları, yapılan çalışmaların değerlendirmesinin yapılabilmesi için önem taşımaktadır ve farklı bilgisayar bilimleri alanlarında da yaygın olarak kullanılmaktadır. Bunun yanında yazılım mühendisliği alanında üretilen uygulamaları daha tutarlı hale getirilebilmesi ve ürünün kalitesi açısından önem taşımaktadır [1]–[6].

Bilgisayar kontrollü sistem teknolojilerinin de gelişimiyle birlikte gerçek zamanlı sistemlerin, hayatın her alanındaki kullanımı artmaktadır ve bu sistemlerin geliştirilmesinde C dilinin yanında Java ve Ada programlama dilleri de kullanılmaktadır. Gerçek zamanlı sistemler belli bir amaç doğrultusunda geliştirilen, güç sınırlamalarıyla uyumlu ihtiyaca yönelik çevresel birimler ile uyumlu çalışan sistemlerdir. Genellikle kullanıcı arayüzü içermeyen bu sistemlerde hata tespiti ve onarımı önemli bir nokta olarak karşımıza çıkmaktadır. Sistemde oluşan hataların tespiti ve hata oluştuğunda ise sistemin görevini münkün olduğunca yerine getirebiliyor olması beklenmektedir. Aynı zamanda gerçekleştirilen görevin sadece tamamlanıyor olması değil “belirli bir zaman içerisinde” tamamlanıyor olması, önem arz etmektedir. Gerçek zamanlı sistemler hakkında daha fazla bilgi almak isteyenler [7] ve [8] numaralı kaynakları inceleyebilirler.

Geliştiriciler gerçek zamanlı sistemlerde programın zamanlama sınırlarını aşmadığından emin olmalıdır. Programın tamamının veya bölümlerinin yürütme süresi hakkında bilgi almak için en kötü yürütme süresi analizi (WCET) gerçekleştirilir. Bu

konuda bilgi almak için [9]–[11] çalışmaları incelenebilir. WCET analizi için geliştirilmiş kıyaslama programları ve kıyaslama uygulama kümeleri mevcuttur. Bu tezde, 2. bölümde WCET hakkında ayrıntılı bilgi verilecektir.

Gerçek zamanlı sistemlerin gelişimde kullanılanılmak üzere geliştirilen kıyaslama uygulamalarının çoğu C dilinde yazılmıştır; kullanılan dil ve platform açısından çeşitlilik göstermemesi bir eksiklik olarak karşımıza çıkmaktadır. Bu çalışmayı oluştururken ana motivasyon konusu bu olmuş ve bu ihtiyaca yönelik Ada dilinde ABench kıyaslama uygulama kümesini (ABench 1.0) oluştuduk. ABench kıyaslama uygulama kümesi Sakarya Üniversitesi Bilgisayar ve Bilişim Bilimleri Fakültesi, Yazılım Mühendisliği Bölümü, Gerçek Zamanlı Sistemler Araştırma Laboratuvarı bünyesinde gerçekleştirilmiştir. Bu kapsamda geliştirilen uygulamalara ve çalışmalara web sitesi üzerinden erişim sağlanabilmektedir [12].

BÖLÜM 2. WCET

Gerçek zamanlı sistemler belirlenen bir amaca yönelik, güç sınırlamaarı ve çevresel birimler ile uyumlu çalışan sistemler olarak tasarlanırlar. Genel olarak kullanıcı arayüzü olmayan sistemlerde oluşan hataların tespiti ve hata oluştuğunda ise sistemin görevini münkün olduğunca yerine getirebiliyor olması önemli bir nokta olarak karşımıza çıkmaktadır. Sistem tarafından gerçekleştirilen işlemin sadece tamamlanıyor olması değil belirlenen zaman içerisinde tamamlanıyor olması gerekmektedir. Gerçek zamanlı sistem geliştiricileri bu noktada programın tamamının veya belirli bir bölümünün yürütme süresi hakkında bilgi almak WCET analizi gerçekleştirilir.

WCET analiz yöntemlerini şu şekilde örneklendirebiliriz; ölçüm temelli analiz, statik analiz ve hibrid tabanlı analiz. Ölçüm temelli analiz, farklı program girdileri ve koşullarında sistemin nasıl davranış sergilediğini kontrol eder, WCET tahminlerini bu bilgiler ile değerlendirir. Statik analiz söz konusu yazılımın ve donanımın matematiksel modellerine dayanır. Donanım modeli, tek tek komutların yürütme süresinin belirlenmesidir. Yazılım modeli, olası yürütme akışlarını temsil eder. Modellerin birleştirilmesi ile bu programda uygulanabilir olan yol, kod parçalarının yürütme sıklığı, vb. WCET tahminine neden olur. Modellerin doğru olması koşuluyla WCET tahmini her zaman güvenlidir. Hibrid ölçüm temelli analiz donanım modeli oluşturmaması dışında statik analize benzer şekilde çalışır. WCET hesaplamasında akış bilgilerini kullanarak bunları küçük program parçalarının yürütme zamanlarını farklı girdilerle oluşturur. Bu analiz yöntemi ölçüm temelli analiz ve hibrit analiz yöntemlerinin karması olarak karşımıza çıkmaktadır [9].

Son zamanlarda WCET analizi için birçok araç geliştirilmiştir. Örneğin OTAWA (Open Tool for Adaptive WCET Analysis) [13], WCET analizindeki katkılar şöyledir,

bazı donanım platformları için WCET hesabı için araç kümesi ve buna ek olarak yeni geliştirilen donanım ve deneysel amaçlarla yeni araçlar geliştirmek için bir framework sunar. Bu araç akış grafikleri, veri akış analizleri, döngü sınırlarının belirlenmesi ve algılanması gibi WCET analizlerini sağlamaktadır. Bu araç ARM, Sparc, TriCore, PowerPC, Star12X gibi bilgisayar mimarilerini de desteklemektedir.

Bir diğer WCET analiz aracı da SWEET (SWEdish Execution Time) isimli araçtır [14]. Bu araç İsveç'te 2001'de bir araştırmacı grubu tarafından geliştirilmiştir. Açık kaynak kodludur. Bu araç basit döngüden programın çeşitli durumlarını hesaplayabilir. Bu aracın oluşturulma amacı akış analizleridir. Programı kullandığı akış yolları, döngüler hakkında bilgi verir.

Diğer bir analiz aracıda RapiTime [15], gelişmiş gömülü mikroişlemcilerde çalışan karmaşık yazılım için çalışan en kötü uygulama sürelerini belirleme sorununa çözüm sunan bir analiz aracıdır. Bunlar dışında geliştirilen birçok WCET analiz aracı mevcuttur.

Bunun yanında WCET analiz yöntemleri ve araçlarının değerlendirilmesi ve kıyaslanması oldukça önemlidir. Bu duruma yönelik çözüm olarak WCET analizleri için geliştirilen programları içeren kıyaslama uygulama kümeleri kullanılır. WCET analizi için çeşitli ihtiyaçları karşılayacak pek çok birçok kıyaslama uygulama kümesi mevcuttur. Bu kıyaslama uygulamaları analiz ihtiyacına göre oluşturulmuştur. Bu konuda kıyaslama uygulama kümeleri hakkında detaylı olarak bilgi bölüm 3'te verilecektir.

BÖLÜM 3. KIYASLAMA UYGULAMALARI

3.1. Kıyaslama Uygulama Kümesi Kavramı

Kıyaslama uygulamaları, bilgisayar sistemlerini, gerçek zamanlı sistemleri, algoritmaları ve bunun yanında yazılım sistemi kullanılan pek çok alandaki çalışmaları kıyaslamak ve değerlendirmek için hem geliştiriciler hem de araştırmacılar tarafından kullanımı mevcut olan uygulamalardır. Kıyaslama uygulamaları, sadece gerçek zamanlı uygulamalar için değil bilgisayar bilimleri alanında ihtiyaç olan her alanda kullanımı mevcuttur. Kullanılan alanda üretilen uygulamaların daha tutarlı hale getirilebilmesi ve ürünün kalitesi açısından önem taşımaktadır.

3.2. Literatürde Yer Alan Kıyaslama Uygulama Kümeleri

Kıyaslama ve değerlendirme işlemi, bilgisayar bilimlerinin her alanında geliştirilen sistemin durumunun ölçeklenebilmesi ve eksikliklerinin giderilebilmesi için önemli ve vazgeçilmesi mümkün olmayan bir işlemdir. Geliştirilen sistemin kullandığı algoritmalar ve her türlü araç, yazılım ve donanım sistemlerine etkilerinin nesnel olarak değerlendirilebilmesi için kıyaslama uygulamaları önem arz etmektedir.

Bilgisayar bilimlerinde için farklı amaçlar için kullanılmak üzere çok sayıda kıyaslama uygulama kümesi geliştirilmiştir. Örneğin, Standart Performans Değerlendirme Kurumu (SPEC-Standard Performance Evaluation Corporation) [6] bulut performansını ölçen ilk benchmark paketi SPEC Cloud_IaaS 2016'ı bulut sağlayıcıları, bulut teknolojisi müşterileri, donanım sağlayıcılar, sanallaştırma yazılımı geliştiricileri, uygulama yazılım sağlayıcıları ve akademik araştırmacıların kullanımını hedeflemektedir. SPEC kıyaslama uygulama kümesi, altyapı olarak (IaaS) kamu ya da

özel bulut platformlarının performanslarının karşılaştırılmasını ele almaktadır. Diğer bir örnek kıyaslama uygulama kümesi, BigDataBench-S'dir [16]. Büyük veri kıyaslama uygulamalarını içerir. Kıyaslama uygulamaları, özellikle, veri depolama, sorgu optimizasyonu ve karmaşık matris hesaplamalarının değerlendirilebilmesi için uygun uygulama parçalarını içerir.

Bizim oluşturmuş olduğumuz kıyaslama uygulamalarının temelini oluşturan fikir ise gerçek zamanlı ve gömülü sistemlerin değerlendirilebilmesi için oluşturulmuş olmasıdır. Bununla alakalı olarakta, gerçek zamanlı sistemler için bilinen referanslardan biri olan Mälardalen WCET kıyaslama uygulamalarıdır [17]. Mälardalen kıyaslama uygulama kümesi özellikle WCET (Worst-case execution time- En kötü durum çalıştırma zamanı) [9] analizine yöneliktir. Mälardalen kıyaslama uygulama kümesi C programlama dilinde yazılmış birçok farklı özellikler ve algoritmalar içeren program parçası içerir.

Gerçek zamanlı ve gömülü sistem geliştiricileri, ilgili araştırmacılar ve bilim adamları oluşturmuş oldukları programların farklı özelliklerini değerlendirebilmeleri ve test etmelerini sağlamak için; dizi işlemleri, döngüler, sıralama algoritmaları ve birçok farklı konuda yazılmış olan uygulamaları içermektedir.

Gerçek zamanlı kıyaslama uygulamalarına diğer bir örnek olarak TACLeBench [3], en kötü durum yürütme zamanı araştırmasına yardımcı olmayı amaçlayan bir dizi değerlendirme programı içermektedir. TACLeBench kıyaslama uygulama kümesi paralel ve paralel olmayan C dili kullanılarak geliştirilmiş 53 adet kıyaslama uygulaması içermektedir.

Gerçek zamanlı uygulamaların çok iş parçacıklı olarak çalışmasını konu alan ve bu alanda en kötü durum yürütme zamanı araştırmasına yardımcı olmayı amaçlayan diğer bir kıyaslama uygulama kümesi de PBench 1.0'dır. Uygulama kümesinde yer alan uygulamalar hem tek iş parçacıklı hem de çok iş parçacıklı olarak yer almaktadır [18].Bunlara ek olarak MiBench [19] kıyaslama uygulama kümesi, ticari olarak temsil edilen bir dizi programı inceler ve bunları mevcut bir benchmark paketi olan

SPEC2000 ile karşılaştırır. Bazı özellikleri bellek davranışı ve kullanılabilir paralellik gibi mevcut SPEC ölçütlerinden ayrılır. Daha detaylı bilgi için [6] incelenebilir.

Literatürde yer alan bazı kıyaslama uygulama kümeleri bu şekildedir. Bu tezde geliştirilen kıyaslama uygulama kümesi ABench 1.0'dir. Geliştirilen kıyaslama uygulama kümesi için Ada programlama dili tercih edilmiştir. Bunun sebebi bu dilde geliştirilen başka kıyaslama uygulama kümesinin bulunmamasıdır. Ada programlama dilinin yapısal özellikleri ve kullanım alanları hakkında Bölüm 4'de detaylı bilgi verilecektir.

Aşagıda gösterilen Tablo 3.1.'de literatürde yer alan bazı kıyaslama uygulama kümelerinin destekledikleri özelikler hakkında özet bilgi verilmiştir.

Tablo 3.1. Kıyaslama Uygulama Kümeleri Özellikleri

Program Adı	Özellikler					
	Paralel Programlama	Praogramlama Dili	Gerçek Zamanlı Programlama Desteği/WCET Analizi	Big Data Optimizasyonu ve Performansı	Gerçek Zamanlı İşletim Sistemi Desteği	Bulut Performans Ölçümü
MiBench	-	C	-	-	-	+
TACLeBench	-	C	+	-	-	-
Mälardalen	-	C	+	-	-	-
BigDataBench-S	-	MongoDB ve HBase	-	+	-	-
SPEC	-	Java	-	-	-	+
PBench	+	C	+	-	+	-
ABench	+	Ada	+	-	-	-

BÖLÜM 4. ADA PROGRAMLAMA DİLİ

Ada, 1980'lerin başında (bu versiyon genellikle Ada 83 olarak bilinir), Fransa'daki CII-Honeywell-Bull'da Dr. Jean Ichbiah tarafından yönetilen bir ekip tarafından geliştirilmiştir. Dil, ABD'deki Intermetrics'ten Bay Tucker Taft'ın öncülüğünde 1990'ların başlarından itibaren revize edildi ve geliştirildi. Sonuç olarak, Ada 95, uluslararası standartlaştırılmış ilk nesne yönelimli dil olarak ortaya çıkmıştır. Stadartlarının belirlenmesi ISO tarafından yapılmıştır, yapılan küçük revizyonlar ile Ada 2005 ortaya çıkmuştır. Dil standardının en yeni sürümü, diğer özelliklerin yanı sıra sözleşme tabanlı programlama (contract based programming) için tam destek sunan Ada 2012'dir [20].

"Ada" ismi bir kısaltma değildir; Charles Babbage ile çalıştığı için dünyanın ilk programcısı olarak kabul edilen bir matematikçi olan Augusta Ada Lovelace (1815-1852) onuruna seçilmiştir.

Ada dili, dünya çapında geliştirme ekiplerinin kritik yazılımlar için kullandığı son teknoloji ürünü bir programlama dilidir mikro çekirdekler, gerçek zamanlı sistemler, büyük ölçekli kurumsal uygulamalara kadar kullanım alanı çısından geniş bir yelpazeye sahiptir.

Ada dilinin kullanım alanlarını şu şekilde örneklendirebiliriz;

- Bordro sistemleri, ticari bankacılık sistemleri, hisse senedi alım satım işlemleri, dil çeviri sistemleri.
- Jeofizik arama ve veri işleme sistemleri ve kimyasal analiz sistemleri.
- Ticari cep telefonu telekomünikasyon uygulamaları.

- Ticari jetlerde, hava trafik kontrol sistemleri, uçuş algılama ve rehberlik sistemleri, uçuş eğitim simülatörleri ve uçuş kontrol / uçuş görüntüleme sistemleri.
- National Aeronautics and Space Administration (NASA)'nın Uzay Mekiği ve Uzay İstasyonu Ortamları.
- Otomatik üretim sistemleri, otomatik malzeme taşıma sistemleri, robotik kaynak sistemleri ve envanter yönetimi sistemleri.
- Gerçek zamanlı tıbbi izleme sistemleri, fotokopi, teksir ürünlerinin gerçek zamanlı dahili kontrolü
- Askeri gömülü sistemler.

Ada programlama dili genel olarak gerçek zamanlı uygulamalarda, sistemlerde kullanılmaktadır.

Ada dilinin mevcut özellikleri aşağıda listelenmiştir.

- Kapsamlı derleme zamanı ve çalışma zamanı kontrolleri
- Nesneye yönelimli programlama
- Çok çekirdekli, eşzamanlı programlama özellikleri
- Generik programgramlama şablonları (Generic templates)
- Kapsülleme
- Hiyerarşik program oluşturma özellikleri

Ada programlama dili, sistem programlama ve gerçek zamanlı sistemler için özel destek sunar. Ve dilin en yeni sürümü, fonksiyonel kontrolleri, dinamik kontroller veya statik analiz ile kaynak kodun bir parçası olarak işlev gören, sözleşmeye dayalı programlama desteği içerir.

Bölüm 5'de Ada programala dili ile geliştirdiğimiz ABench kıyaslama uygulama kümesi hakında detaylı bilgi verilecektir.

BÖLÜM 5. ABENCH

5.1. Ada Dilinde, Kıyaslama Uygulama Kümesi

Bu tez çalışmasında Ada programlama dilinde gerçek zamanlı kıyaslama uygulama kümesinin ilk versiyonu olan ABench 1.0'ı geliştirdik. Kısaltma olarak ABench olarak isimlendirdiğimiz kıyaslama uygulama kümesinin açılımı Ada Benchmark'dır. Çalışmamız açık kaynak kodlu, serbest kullanım ve geliştirmeye açıktır. ABench 1.0'daki uygulamalara ve çalışma detaylarına web sayfamız üzerinden erişim sağlayabilirsiniz [12].

ABench'in tasarım sürecinde ilk olarak WCET analizi ile ilgili benchmark çalışmalarını inceledik ve bu alanda ihtiyaç duyulan iyileştirmeleri araştırdık. Malardalen Benchmark makalesi, WCET analiziyle ilgili kriterler üzerinde yapılması gereken çalışmalar hakkında incelemlerde bulundugumuzda İhtiyaçlardan biri, C ++, Java ve Ada gibi C programlama dili dışındaki farklı programlama dillerinin desteğiydi. Kıyaslama uygulama kümesini Ada programlama dilinde geliştirmeye karar verdik.

İkinci aşamada, destekleyecek işletim sistemini belirledik. Linux dağıtımlarını özgürce erişilebilir, açık kaynaklı ve araştırmacılar tarafından yaygın olarak kullanıldığı için seçimimizi bu yönde yaptık.

Üçüncü aşama olarak, programların özellikleri seçim aşamasıdır. Bu aşamada kıyaslama uygulama kümesi tarafından desteklenecek programların özellikleri belirlendi. Özellik seçimi, tasarım sürecinin önemli bir parçasıdır, çünkü değerlendirmede kıyaslama uygulama kümesi kullanımı sırasında, araştırmacılar araçlarının / yöntemlerinin davranışını farklı programlama özelliklerine göre test

etmek istemektedir. Bu duruma yönelik olarak seçimlerimizi çeşitlendirerek temel program özeliklerini destekleyecek özellikler belirledik bu özellikler Tablo 5.1.'de detaylı olarak açıklanmıştır.

Dördüncü aşamada kıyaslama programlarında uygulanacak algoritmaları belirledik. Bu aşamada, uygulamalarını daha önce belirlenen özelliklerin kullanımını destekleyecek şekilde seçtik. Her program için bilinen bir algoritma seçimi yaptık

ABench kıyaslama uygulama kümesinde yer alan uygulamaların her biri Ada programlama dilinde yazılmış olan uygulama paçacıklarıdır. ABench kıyaslama uygulama kümesinde toplam 11 uygulama yer almaktadır. Ada dili gerçek ve gömülü sistem gelişiminde kullanılan dillerden biri oldugundan çalışmamızda yer alan uygulamaların tamamı Ada programlama dili kullanılarak oluşturuldu.

5.1.1. Özellik matrisi

ABench kıyaslama uygulamalarını tasarlarken, faklı alanlardaki kullanım ihtiyacı olabilecek temel konuları belirledik, bu konularda geliştireceğimiz uygulamaları seçtik ve ada programlama dilinde kodlanmasını sağladık. Kodlanan her program parçacığının özelliklerini uygulama kümesinin genelinde yer alan uygulama özellik matrisinde açıkladık. Bu matris yazmış oldugumuz uygulamaların belirlemiş oldugumuz temel konulardan hangilerini içerdiğini göstermektedir. ABench kıyaslama uygulamalarının özellik matrisi Tablo 5.2.'de gösterilmektedir. Tabloda program parçalarının desteklediği özellikler artı (+), desteklenmeyen özellikler ise eksi ile (-) gösterilmektedir.

Özellik matrisimizde yer alan ana başlıklar Tablo 5.1.'de detaylı bir şekilde açıklanmıştır. Rahat kullanım açısından başlıklar kısaltmalar ile kullanılmaktadır.

Tablo 5.1. Program özellik açıklamaları

Özellik Adı	**Kısaltma**	**Açıklama**
Single-threaded	ST	Tek iş parçacıklı programlardır.
Multi-threaded	MT	Çok iş parçacıklı programlardır.
External Routine	ER	Harici kütüphane kullanan programlarıdır.
Singlepath	SP	Programın akışı, her zaman aynı yürütme yolunu izler.
Multi path	MP	Program her bir akış durumunda farklı bir yürütme yolunu takip edebilir.
Dynamic Memory	DM	Program, program verileri için belleği dinamik olarak ayırır ve serbest bırakır.
Loop	L	Döngü yapıları içerir.
Nested Loop	NL	İçiçe döngü yapıları içerir.
Recursion	R	Program özyinelemeli fonksiyon içerir.
Decision	D	Program karar yapıları içerir.
Array	A	Program dizi yapısı içerir.
Bit Level Operation	BLO	Program bit düzeyinde işlemler içerir.
Floating Point Operation	FPO	Program kayan noktalı sayı tipi içeren işlemler gerçekleştirir.
Integer Operation	IO	Program tam sayı tipi içeren işlemler gerçekleştirir.
InputVector	IVEC	Program, parametre olarak vektor tipinde girdi alır.
Input Value	IVAL	Program parametre olarak tek bir giriş değeri alır.
Input File	IF	Program parametre olarak bir dosyadan veri alır.
File Input/Output	FileIO	Program dosyadan girdi okur ya da dosyaya çıktı yazar.

Bilgisayar bilimleri, matematik, olasılık, optimizasyon gibi alanlardaki bazı sorunlara yönelik geliştirilmiş algoritmaları seçerek bu sorunları Ada programlama dilini kullanarak kodlanmıştır ve yukarıda belirtilen özellikler açısından incelenmesini sağlanmıştır. Seçilen problemler sıralama, arama, matris çarpımı, korelasyon hesaplama, genetik algoritma (optimizasyon), ağaç yapılarında dolaşma (tree traversal), bitwise operatörlerin kullanımı algoritmalarıdır. Geliştirdiğimiz programlar

literatürde yer alan isimleriyle kıyaslama uygulama kümesinde yer almaktadır. Geliştirdiğimiz kıyaslama uygulama kümesi ile alakalı özellik matrisi Tablo 5.2.'de gösterilmektedir.

Tablo 5.2. Program özellik matrisi

Program Adı	**Özellikler**																	
	ST	MT	ER	SP	MP	DMA	L	NL	D	A	BLO	R	FPO	IO	IVEC	IVAL	IF	FileIO
BubbleSort	+	-	+	+	-	-	-	+	+	+	-	-	-	+	-	-	-	-
SelectionSort	+	-	+	+	-	-	+	+	+	+	-	-	-	+	-	-	-	-
QuickSort	+	-	+	+	-	+	+	+	+	+	-	+	+	-	-	-	-	-
Fibonacci	+	-	+	-	+	-	-	-	+	-	-	+	-	+	-	-	-	-
BinarySearch	+	-	+	-	+	+	+	-	+	+	-	-	-	+	-	-	-	-
MatrixMultiplication	+	-	+	+	-	-	+	+	-	-	-	-	+	-	-	-	-	-
CorrelationCalculation	+	-	+	+	-	-	+	-	-	+	-	-	+	-	-	-	-	-
TreeTraversal	+	-	+	+	-	+	-	-	+	-	-	+	-	+	-	-	-	+
Evolution	+	-	+	+	-	-	+	+	+	+	-	-	+	+	-	-	-	-
EvolutionFI	+	-	+	-	+	-	+	+	+	+	-	-	+	+	-	-	+	+
BitwiseOperations	+	-	+	+	-	-	-	-	-	-	+	-	-	-	-	-	-	-

5.2. Kıyaslama Uygulamalarının Detayları

Bu bölümde kıyaslama uygulamaları hakkında ayrıntılı bilgi veriyoruz. Kıyaslama uygulamaları literatürde varolan bilinen algoritmaların Ada programlama dilinde kodlanmış şeklidir. Bu bölümde bu algoritmalar ile alakalı detaylara yer verilecektir.

- BubleSort, bu program kabarcık sıralama algoritmasını kullanarak bir dizinin elemanlarını azalan düzene göre sıralar. Diziyi oluşturan ilk değerleri program içerisinde doğrudan yer alamaktadır.
- SelectionSort, bu program seçimli sıralama algoritmasını uygulayarak artan sıralı bir dizi oluşturur.

- QuickSort, bu program hızlı sıralama algoritmasını kullanarak kesirli sayıları artan sıralı olarak sıralar.
- Fibonacci, bu program komut satırından girilen pozitif bir tamsayı girdisi için fiboacci hesabı yapar.
- BinarySearch, bu programda kod içerisinde tanımlanmış bir sayı dizisi içersinde komut satırından girilen bir değerin kaçıncı indekste olduğunu ikili arama algoritmasıyla hesaplar.
- MatrixMultiplication, kod içerisnde var olan iki matrisin çarpımını ayrı bir matrise yazar.
- CorelationCalculation, bu program kod içerisinde float olarak tanımlanmış 2 dizi için korelasyon hesabını yapar.
- TreeTraversal, bu program kod içerisinde tanımı yapılmış bir ağaç yapısında düğümler arasında iç tertip (inorder), ön tertip (preorder) ve son tertip (postorder) yöntemlerini kullanarak gezinmesi sonucunda sırayla uğradığı düğümleri dosyaya çıktı olarak üretir.
- Evolution, bu programda başlangıçta harfleri değiştirlmiş anlamsız bir cümle için genetik algoritma kullanılarak en uygun değere yaklaşımını sağlar, optimizasyon algoritması oldugu için verilen girdinin karmaşıklığı ve uzunluğu arttıkça sonuç süresi değişebilmektedir.
- EvolutionFI programı da işleyiş olarak Evolution ile aynı olup girdiyi bir dosyadan okumaktadır.
- BitwiseOperations, program içerisinde statik olarak tanımlanan sayının 32 bit düzenine çevirir ve sağa, sola shift ve and işlemleri yapılmasını sağlar.

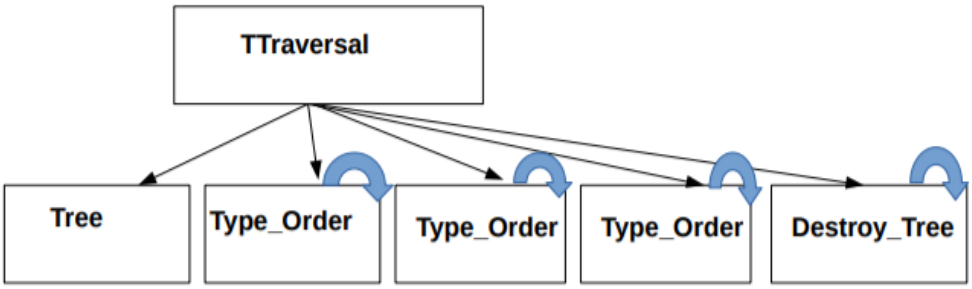
Uygulamalarımızın tamamını Ada programlama dilinde geliştirildi ve GNATLS 5.5.0 sürümünü kullanarak mevcut uygulamalar derlenmiştir.

5.2.1. Uygulamalar hakkında dizin yapılandırması

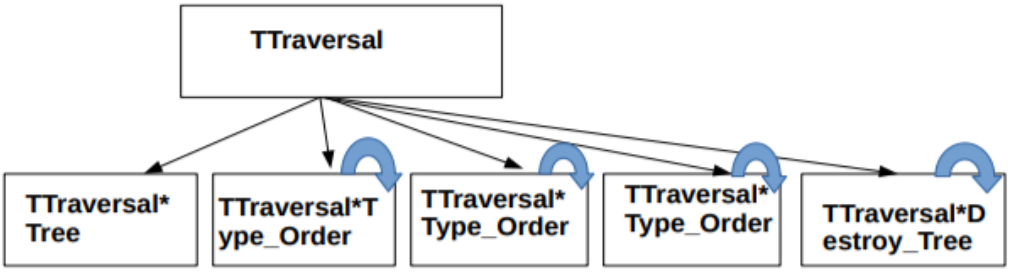
Kıyaslama uygulama kümemizde yer alan her uygulamanın kendine özgü bir dizinde yer almaktadır. Her dizinde standart olarak aşagıdaki bilgiler bulunmaktadır.

- Programın kaynak kodu,
- Makefile dosyası, programın derlenme sürecine yardımcı olmak için,
- Çağrı grafiği (Call graph), programın akışını kolayca anlamaya yardımcı olmak için programın bir PDF dosyası olarak akış grafiği,
- Kapsam hiyerarşi grafiği (Scope hierarchy graph), program akışını, yapısını ve döngü girişlerini anlamaya yardımcı olmak için bir PDF doyası olarak akış ve yapısal analiz grafiği.

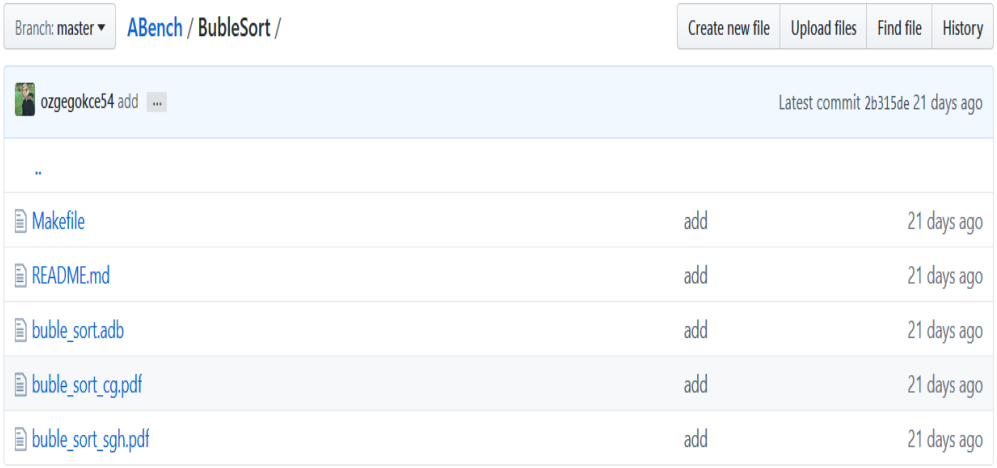
Örnek olması açısında TreeTraversal programına ait çağrı grafiği Şekil 5.1.'de ve kapsam hiyerarşi grafiği de Şekil 5.2.'de gösterilmektedir. Ayrıca Şekil 5.3'de program dizin yapısı gösterilmektedir.



Şekil 5.1. TreeTraversal programı için çağrı grafiği



Şekil 5.2. TreeTraversal programı için kapsam hiyerarşi grafiği



Şekil 5.3. Dizin yapısı

Yukarıdaki Şekil 5.3.'de standart bir dizin yapılanması görülmektedir. Dosyadan girdi alan ya da çıktı üreten programlarda bu dosyalara ek olarak girdi ve çıktı dosyaları da yer almaktadır.

BÖLÜM 6. TARTIŞMA

Bu tezde WCET analiz araçları/yöntemlerinin değerlendirilmesine ve karşılaştılmasına yardımcı olmak amacıyla Ada dilinde geliştirilen ABench kıyaslama uygulama kümesi oluşturulmuştur. Bu uygulama kümesinin en önemli özelliği geliştirilen kıyaslama uygulama kümelerinden farklı olarak Ada programa dili kullanılarak geliştirilmiş olmasıdır. Gerçek zamanlı sistemlerin geliştirilmesi esnasında WCET analiz çalışmalarına yardımcı olan uygulamalar genellikle C programlama dili kullanılarak geliştirilmiştir. Bu eksikliğin giderilmesine yönelik bu çalışma gerçekleştirilmiştir.

Ada dili yapısal anlamda kullanıcılara birçok konuda destek sunmaktadır. Birçok işlem mevcut kütüphaneleri sayesinde kolay bir şekilde yapılabilmektedir. Örneğin matris çarpım işlemi matrisler tanımlandıktan sonra 2 sayının çarpımı gibi yapılabilmektedir. Dil aynı zamanda genel tip desteği sayesinde tipe bağımlı olmadan nesne yönelimli programlamayı desteklemektedir. Nesne yönelimli programlamayı desteklemesi sürekli büyümekte olan yazılım sistemleri için karmaşıklığın azaltılması, tekrar kullanılabilirliğin artırılması ve pekçok konuda önem taşımaktadır.

Ada programlama dili gelişmiş yazılım sistemleri için tasarlanmıştır. Ada prosedürel yapıda bir dil olmasından kaynaklı olarak prosedürler ayrı ayrı derlenebilir, bu sayede yürütme başlamadan önce kurulum aşamasında problemlerin tespitini kolaylaştırmaktadır. Prosedürel olmasının ayrı bir avantajı ise derleme zamanında farklı dillerde yürütme zamanında oluşabilecek tespit edilemeyen hataların azaltılmasına yardımcı olması için önemli bir özelliktir. Genellikle kritik yazılım sistemleri olan uzay, savunma sistemlerinde kullanılması sebebiyle bu özellikler önem arz etmektedir.

BÖLÜM 7. SONUÇ

Gerçek zamanlı yazılımların geliştirilmesinde uygulamanın en kötü durum yürütme süresinin hesaplanabilmesi için WCET analizlerine ihtiyaç duyulmaktadır. WCET analizi ayrıca hem endüstride hem de akademik çalışmalarda aktif olarak kullanılmaktadır. WCET analiz araçları / yöntemleri geliştiren araştırmacılar, çalışmalarını alternatiflerle değerlendirmek ve karşılaştırmak için kıyaslama uygulamalarına ihtiyaç duyarlar.

Gerçek zamanlı sistemlerin gelişimde kullanılanılmak üzere geliştirilen kıyaslama uygulamaları C dilinde yazılmıştır ve kullanılan dil, platform açısından çeşitlilik göstermemesi bir eksiklik olarak karşımıza çıkmaktadır. Bu çalışmayı oluştururken ana motivasyon konusu bu olmuş ve bu ihtiyaca yönelik Ada dilinde ABench kıyaslama uygulama kümesini geliştirilmesi sağlanmıştır.

Bu tez çalışmasında gerçek zamanlı sistemlerin WCET analizine yardımcı olmak için için Ada programlama dilinde bir kıyaslama uygulama kümesi geliştirilmesi sağlanmıştır. ABench ismini verdiğimiz kıyaslama uygulama kümesi tamamen açık kaynak kodludur. Gerçek zamanlı sistem geliştiricileri ve bilim insanları oluşturulan bu kıyaslama uygulama kümesini geliştirdikleri sistemlerin testinde kullanabilirler.

Bu çalışma Ada programlama dilinde geliştirilen sistemlerin testi konusunda fayda sağlanaması planlanmıştır. Bu sayede WCET analiz araçlarının testinde dil konusundada çeşitlilik sağlanacaktır. Bu çalışmadan yola çıkarak dil opsiyonunun WCET analizi konusundaki etkisinin incelenmesi sağlanabilecektir.

KAYNAKLAR

[1] F. Nemer, H. Cassé, P. Sainrat, J.-P. Bahsoun, and M. De Michiel, "PapaBench: a Free Real-Time Benchmark," in *6th International Workshop on Worst-Case Execution Time Analysis (WCET'06)*, 2006, vol. 4.

[2] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," *2001 IEEE Int. Work. Workload Charact. WWC 2001*, pp. 3–14, 2001.

[3] H. Falk *et al.*, "TACLeBench: A Benchmark Collection to Support Worst-Case Execution Time Research," *16th Int. Work. Worst-Case Exec. Time Anal. (WCET 2016)*, vol. i, no. 2, p. 10, 2016.

[4] http://www.mrtc.mdh.se/projects/wcet/benchmarks.html., Erişim Tarihi: 19.03.2019.

[5] https://www.rapitasystems.com., Erişim Tarihi: 19.03.2019.

[6] https://www.spec.org., Erişim Tarihi: 19.03.2019.

[7] G. C. Buttazzo, *Hard Real-Time Computing Systems*, vol. 24. Boston, MA: Springer US, 2011.

[8] H. Kopetz, *Real-Time Systems*. Boston, MA: Springer US, 2011.

[9] J. Abella *et al.*, "WCET analysis methods: Pitfalls and challenges on their trustworthiness," in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2015, pp. 1–10.

[10] J. Abella, D. Hardy, I. Puaut, E. Quinones, and F. J. Cazorla, "On the comparison of deterministic and probabilistic WCET estimation techniques," in *Proceedings - Euromicro Conference on Real-Time Systems*, 2014, pp. 266–275.

[11] S. Altmeyer and R. I. Davis, "On the correctness, optimality and precision of Static Probabilistic Timing Analysis," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, 2014, pp. 1–6.

[12] https://rtsrlab.sakarya.edu.tr/index_en.html., Erişim Tarihi: 16.04.2019.

[13] http://www.otawa.fr/., Erişim Tarihi: 19.03.2019.

[14] http://www.mrtc.mdh.se/projects/wcet/sweet/index.html., Erişim Tarihi: 19.03.2019.

[15] http://www.artist-embedded.org/docs/Tools_Platforms/CompilersTA/RapiTime/Rptme_WrstCse_toolkit.pdf., Erişim Tarihi: 19.03.2019.

[16] "B IG D ATA AND AI B ENCHMARK S UITE BigDataBench : A Dwarf-based Big Data and AI Benchmark."

[17] J. Gustafsson, A. Betts, A. Ermedahl, and B. Lisper, "The mälardalen WCET benchmarks: Past, present and future," in *10th International Workshop on Worst-Case Execution Time Analysis, WCET 2010*, 2010, vol. 15, pp. 136–146.

[18] S. Serttaş and V. H. Şahin, "PBench: A Parallel, Real-Time Benchmark Suite," *Acad. Perspect. Procedia*, vol. 1, no. 1, pp. 178–186, Nov. 2018.

[19] http://vhosts.eecs.umich.edu/mibench/., Erişim Tarihi: 19.03.2019.

[20] https://www.adacore.com/about-ada., Erişim Tarihi: 16.04.2019.

ÖZGEÇMİŞ

Özge GÖKÇE, 10.07.1992'de Sakarya'da doğdu. İlk, orta ve lise eğitimini Sakarya'da tamamladı. 2010 yılında Tes-iş Adapazarı Anadolu Lisesi'nden mezun oldu. 2010 yılında başladığı Fatih Üniversitesi Bilgisayar Mühendisliği Bölümü'nü 2014 yılında bitirdi. 2015 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine başladı. 2015 yılında Erguvan Bilişim A.Ş.'de Yazılım Geliştirme Uzmanı olarak 2018 Temmuz ayına kadar çalıştı. 2018 Eylül ayından Nisan 2019' a kadar Digilera Bilgi Teknolojileri Ltd. Şti.'de Yazılım Geliştirme Uzmanı olarak görev aldı. Nisan 2019'dan bu yana Vizyoneks Bilgi Teknolojileri A.Ş.'de Yazılım Geliştirme Uzmanı olarak görev almaktadır.