BSM 420 – BİLGİSAYAR SİSTEMLERİNİN PERFORMANS DEĞERLENDİRMESİ

3.Hafta: Kıyas Setleri (Benchmark sets)

Kıyas Testleri (Benchmarks)

- Performans en iyi gerçek bir uygulama çalıştırılarak belirlenir
  - beklenen iş yüküne özgü programları kullanma
  - veya tipik uygulama sınıfları,

  örneğin derleyiciler / editörler, bilimsel uygulamalar, grafikler vb.

- Küçük testler
  - mimarlar ve tasarımcılar için güzel
  - standartlaştırması kolay
  - istismar edilebilir!

- Karşılaştırma paketleri
  - Uygulama kod setleri
  - Livermore Döngüleri: 24 döngü çekirdeği
  - Linpack: Doğrusal Cebir Paketi
  - SPEC: endüstri organizasyonlarından kodlar

Sistem Performansı Değerlendirme Ortaklığı (SPEC-System Performance Evaluation Corporation)

- Endüstri tarafından desteklenen ancak bağımsız ve kendi kendini yöneten - kod geliştiricileri ve makine satıcıları tarafından güvenilir
- Test için açık kılavuzlar, bkz. www.spec.org
- Düzenli güncellemeler (testler kaldırılır ve alaka düzeyine göre periyodik olarak yenileri eklenir)
- Belirli uygulama sınıfları için özel testler

SPEC Tarihi

- İlk Dönem: SPEC CPU89
  - 10 adet program tek bir sayı üretiyor
- İkinci Dönem: SPEC CPU92
  - SPEC CINT92 (6 adet tamsayı programı) and SPEC CFP92 (14 kayan nokta programı)
  - Derleyici bayrakları farklı programlarda farklı yerlere konabilir
- Üçüncü Dönem: SPEC CPU95
  - Yeni program setleri: SPEC CINT95 (8 tamsayı programı) ve SPEC CFP95 (10 kayan noktalı)
  - Tüm programlar için tek bir bayrak düzeni
- Dördüncü Dönem : SPEC CPU2000
  - Yeni program setleri: SPEC CINT2000 (12 tamsayı programı) ve SPEC CFP2000 (14 kayan noktalı)
  - Tüm programlar için tek bir bayrak düzeni
  - C, C++, Fortran 77, ve Fortran 90 programları
- **SPECviewperf® 13**
- **SPEC CPU 2017**
- **SPEC Cloud IaaS 2018**

Özel SPEC Kıyas Kümeleri

- I/O
- Ağ
- Graphik
- Java
- Web server
- Transaction processing (databases)

Performans Ölçümü

- Bir kıyaslama kümesinden 25 program düşünün - 25 programın davranışlarını tek bir sayıyla nasıl ifade ederiz?

	P1	P2	P3…… P25
Sys-A	?	?	? ?
Sys-B			
Sys-C			
……			

- Çalışma zamanları toplamı (AO-Aritmetik Ortalama)
- Ağırlıklı Çalışma zamanları Toplamı (AO)
- Çalışma zamanlarının geometrik ortalaması (GO)
- Çalışma Zamanlarının harmonik Ortalaması(HO)

ÖRNEK V

- Bir kıyas testinde 3 program düşünelim. Sistem A'nın referans makine olduğunu varsayın. B sisteminin performansı, C sisteminin performansı ile nasıl karşılaştırılır (tüm 3 ölçüm için)?

	P1	P2	P3
Sys-A	5	10	20
Sys-B	6	8	18
Sys-C	7	9	14

ÖRNEK V - ÇÖZÜM

- Bir karşılaştırma testinde 3 program düşünelim. Sistem A'nın referans makine olduğunu varsayın. B sisteminin performansı, C sisteminin performansı ile nasıl karşılaştırılır (tüm 3 ölçüm için)?

	P1	P2	P3	Ç.Z.T.	A.Ç.Z.T	GO
Sys-A	5	10	20	35	3	10
Sys-B	6	8	18	32	2.9	9.5
Sys-C	7	9	14	30	3	9.6

- C ye göre, B 1.03 hız artışına sahip (A.Ç.Z.T) - hızlanma veya 1.01 (GO) veya 0.94 (Ç.Z.T)
- C'ye göre, B 'nin çalışma zamanı 3.3% (A.Ç.Z.T) kadar düşük veya 1% (GO) veya 6.7% (Ç.Z.T)

ÖRNEK VI (Ağırlıklı Çalışma Zamanları Toplamı)

- Bir X referans makinesi iyileştirilir ve üzerinde A, B, C, D programları 1 saniye boyunca çalıştırılır.

- Aynı iş yükü (X üzerindeki ile aynı sayıda komutla dört program yürütülür) yeni bir Y makinesinde çalıştırılır ve Her program için çalışma süreleri 0.8, 1.1, 0.5, 2'dir.

$$\frac{\frac{1}{4}}{\frac{1}{4.4}} = 1.1$$

- Verilen iş yükü için çalışma sürelerinin aritmetik ortalaması açısından, Y'nin X'ten 1,1 kat yavaştır

ÖRNEK VII (Geometrik Ortalama - GO)

	Computer-A	Computer-B	Computer-C
P1	1 sec	10 secs	20 secs
P2	1000 secs	100 secs	20 secs

GO sonuçları : (i) A=B
(ii) C ~1.6 kat daha hızlı

- (i) ye göre P1 P2 ye göre 100 kat daha hızlı çalışmalı
- Bu varsayıma göre, (ii) doğru olamaz

Bu nedenle GO tutarsızlığa neden olur

SONUÇ

- GO: bir referans makine gerektirmez, ancak performansı doğru olarak hesaplamaz
  - Dolayısıyla çalışma zamanları çarpılır ve sys-A nın sysy-B den 1.2x kat daha hızlı olduğu belirlenir.
- AO: belirli bir işyükü için performansı hesaplar, ancak işyükü belirli bir referans makine üzerinde programların çalıştırılmasıyla belirlenir.
  - Belirli periyotlarla, referans makine güncellenmelidir.

ÖRNEK VIII

- Yeni dizüstü bilgisayarımın IPC'si eski dizüstü bilgisayarımdan% 20 daha kötüdür. Ayrıca eski dizüstü bilgisayardan% 30 daha yüksek bir saat hızına sahiptir. Her iki makinede de aynı ikili dosyaları çalıştırıyorum.
- Yeni dizüstü bilgisayarım ne kadar hızlı?

ÖRNEK VIII

- Yeni dizüstü bilgisayarımın IPC'si eski dizüstü bilgisayarımdan% 20 daha kötüdür. Ayrıca eski dizüstü bilgisayardan% 30 daha yüksek bir saat hızına sahiptir. Her iki makinede de aynı ikili dosyaları çalıştırıyorum.
- Yeni dizüstü bilgisayarım kaç kat hızlıdır?

Çalışma zamanı= çevrim zamanı* CPI * komut
Performans = saat hızı * IPC / komut
Hızlanma = yeni perf/ eskiperf
= yeni saat hızı* yeni IPC / eski saat hızı* eski IPC
= 1.3 * 0.8 = 1.04

ALTERNATİF BAKIŞ I

- Her programın eşit sayıda çevrimde çalıştığı varsayılır

- Çevrim başına yürütülen talimat sayısı (IPC), bir programın bir sistemde ne kadar iyi performans gösterdiğinin bir ölçüsüdür

- Yaklaşık sonuç IPC'lerin toplamı veya

  IPC Aritmetik Ortalaması = $\frac{1.2 \text{ instr}}{\text{cyc}} + \frac{1.8 \text{ instr}}{\text{cyc}} + \frac{0.5 \text{ instr}}{\text{cyc}}$

- Bu ölçüm A programı içindeki 1 talimatın B programındaki 1 talimat ile aynı öneme sahip olduğunu gösterir.

ALTERNATİF PERSPEKTİF - II

- Her programın eşit sayıda talimatla çalıştığı varsayılır.

- Talimat başına gereken çevrim sayısı(CPI), bir programın bir sistemdeki performansının ölçüsüdür

- Uygun bir ölçüm CPI toplamı ve ortalamasıdır

$$= \frac{0.8 \text{ cyc}}{\text{instr}} + \frac{0.6 \text{ cyc}}{\text{instr}} + \frac{2.0 \text{ cyc}}{\text{instr}}$$

- Bu ölçüm , prog-A'daki 1 talimatın prog-B'deki 1 talimat ile aynı öneme sahip olduğunu varsayar.

AO ve HO

$$H.O = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3} + \ldots + \frac{1}{a_n}} \text{ dir.}$$

- $IPC_{AO} = 1 / CPI_{HO}$
  $CPI_{AO} = 1 / IPC_{HO}$

HO: Harmonik Ortalama

- Dolayısıyla, bir kıyas paketindeki programlar, her biri eşit sayıda çevrim için çalışacak şekilde ağırlıklandırılırsa, IPC'lerin AO'sı veya CPI'ların HO'sı her ikisi de uygun ölçütlerdir.

- Bir kıyas paketindeki programlar, her biri eşit sayıda talimat için çalışacak şekilde ağırlıklandırılırsa, CPI'ların AO'sı veya IPC'lerin HO'sı her ikisi de uygun ölçütlerdir.

AO vs. GO

- $IPC_{AO} = 1 / CPI_{GO}$

- IPC'lerin AO'sı, her bir programın her bir çevriminin sırayla çalıştığı bir iş yükü çıkışına(throughput) eşittir; ancak yüksek IPC'li programlar AO'ya daha fazla katkıda bulunur

- IPC'lerin GO'si herhangi bir gerçek iş yükü için çalışma zamanını göstermez; ancak her programın IPC'si nihai ölçüme eşit katkıda bulunur

ÖRNEK IX

- Yeni dizüstü bilgisayarım, eski dizüstü bilgisayarımdan % 30 daha yüksek bir saat hızına sahip. Her iki makinede de aynı ikili dosyaları çalıştırıyorum. IPC'leri aşağıda listelenmiştir. İkili dosyalar eşit CPU zamanı alacak şekilde çalışıyor.
- Yeni dizüstü bilgisayarım kaç kat hızlanmıştır?

	P1	P2	P3
Eski-IPC	1.2	1.6	2.0
Yeni-IPC	1.6	1.6	1.6

ÖRNEK IX

- Yeni dizüstü bilgisayarım, eski dizüstü bilgisayarımdan % 30 daha yüksek bir saat hızına sahip. Her iki makinede de aynı ikili dosyaları çalıştırıyorum. IPC'leri aşağıda listelenmiştir. İkili dosyalar eşit CPU zamanı alacak şekilde çalışıyor.
- Yeni dizüstü bilgisayarım kaç kat hızlanmıştır?

	P1	P2	P3	AO	GO
Eski-IPC	1.2	1.6	2.0	1.6	1.57
Yeni-IPC	1.6	1.6	1.6	1.6	1.6

IPC Aritmetik Ortalaması doğru ölçümdür. GO da kullanılabilir.
AO kullanılarak hızlanma 1.3 kat olur

Bilgisayar Tasarımı Prensipleri

- Eş zamanlılıktan faydalanma
  - Çok-işlemcili sistemler, diskler, bellekler, pipeline, çok fonksiyonlu birimler
- Yerellik prensibi
  - Veri ve talimatların yeniden kullanımı
- Ortak duruma odaklanmak
  - Amdahl Kuralı

Amdahl Kuralı

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

ÖRNEK

- Webde kullanılan bir işlemciyi hızlandırmak istiyoruz. Yeni işlemci 10 kat eskisine göre daha hızlı. Eski işlemci %40 işleme ve % 60 I/O ile meşgul. Hızlanma nedir?

ÖRNEK

- Webde kullanılan bir işlemciyi hızlandırmak istiyoruz. Yeni işlemci 10 kat eskisine göre daha hızlı. Eski işlemci %40 işleme ve % 60 I/O ile meşgul. Hızlanma nedir?

$$\text{Fraction}_{\text{enhanced}} = 0.4;$$

$$\text{Speedup}_{\text{enhanced}} = 10;$$

$$\text{Speedup}_{\text{overall}} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

ÖRNEK

- Grafik işlemcilerinde gerekli ortak bir dönüşüm kare kökü işlemi ile ilgilidir. Kayan noktalı (FP) kare kök uygulamaları, özellikle grafikler işlemciler için performans açısından çok önemlidir. FP kare kökü işleminin (FSQRT), kritik bir grafik kıyas setinde çalışma süresinin % 20'sinden sorumlu olduğunu varsayalım. Hedef FSQRT donanımını geliştirmek ve bu işlemi 10 kat hızlandırmaktır.
- Diğer alternatif, grafik işlemcideki tüm FP talimatlarını 1,6 kat daha hızlı çalıştırmaktır; FP talimatları, uygulamanın yürütme süresinin yarısından sorumludur. Tasarım ekibi, tüm FP talimatlarını hızlı kare kök işlemi ile 1,6 kat daha hızlı çalıştırabileceklerine inanıyor.
- Bu iki tasarım alternatifini karsılastırın.

$$\text{Speedup}_{\text{FSQRT}} = \frac{1}{(1-0.2)+\frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$

$$\text{Speedup}_{\text{FP}} = \frac{1}{(1-0.5)+\frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$

ÖRNEK

- Bir programın bir makinede 100 saniyede çalıştığını ve bu sürenin 80 saniyesinin çarpma işlemi olduğunu varsayalım.
- Programın 4 kat daha hızlı çalışmasını istiyorsak çarpma hızını ne kadar artırmamız gerekir?
- 5 kat daha hızlı için ne olur?

ÖRNEK

- Tüm kayan nokta talimatlarını 5 kat daha hızlı çalıştıracak bir makineyi geliştirdiğimizi varsayalım. Kayan nokta geliştirme işleminin öncesinde bazı kıyas setlerinin çalışma süresi 10 saniyedir.
- 10 saniyenin yarısı kayan nokta talimatlarını uygulamak için harcanırsa hızlanma ne olur?

ÖRNEK

Bir önceki örnekteki yeni kayan nokta biriminin performansını ölçmek için bir kıyas seti arıyoruz ve genel kıyaslama setinin 3 kat hızlandığını göstermek istiyoruz. Bir kıyas seti, eski kayan nokta donanımıyla 100 saniye boyunca çalışıyor.

- Bu kıyaslamada istediğimiz hızlandırmayı sağlamak için kayan nokta komutlarının yürütme süresinin ne kadarı bu programda dikkate alınır?

ÖRNEK

- Kayar noktalı aritmetik birimi 2 kat hızlandırılsa, fakat tüm komutların ancak %10'u kayar noktalı aritmetik olsa toplam hızlanma ne kadar olur?

ÖRNEK

- Kayar noktalı aritmetik birimi 2 kat hızlandırılsa, fakat tüm komutların ancak %10'u kayar noktalı aritmetik olsa toplam hızlanma ne kadar olur?

$$\text{ExTime}_{new} = \text{ExTime}_{old} \times (0.9 + .1/2) = 0.95 \times \text{ExTime}_{old}$$

$$\text{Speedup}_{overall} = \frac{1}{0.95} = 1.053$$

ÖRNEK

- Bir disk alt sisteminin hata oranlarının oranı

$$\text{Failure rate}_{\text{system}} = 10 \times \frac{1}{1{,}000{,}000} + \frac{1}{500{,}000} + \frac{1}{200{,}000} + \frac{1}{200{,}000} + \frac{1}{1{,}000{,}000}$$

$$= \frac{10+2+5+5+1}{1{,}000{,}000 \text{ hours}} = \frac{23}{1{,}000{,}000 \text{ hours}}$$

- İyileştirilebilecek hata oranı fraksiyonu milyon saat başına 5 dir veya 0,22 dir
- Güvenilirlik iyileştirmesi nedir?

$$\text{Improvement}_{\text{power supply pair}} = \frac{1}{(1-0.22) + \frac{0.22}{4150}} = \frac{1}{0.78} = 1.28$$