PARALEL GÖZLEMLEME (MONİTÖR) SİSTEM MİMARİSİ

Ahmet ÖZMEN

Elektrik-Elektronik Bölümü
Mühendislik Fakültesi
Dumlupınar Üniversitesi, 43000, Kütahya
e-posta: ozmen@dumlupinar.edu.tr

Anahtar sözcükler: Paralel Gözlemleme (Monitör) Sistemleri, Paralel Performans

ABSTRACT

This paper presents a minimally intrusive monitoring architecture for parallel performance tuning and debugging. The architerture creates a common platform to address monitoring concepts and problems. It is not restricted to a specific monitoring system, and provides a means to enable comprehensive analysis and validation. Using this architecture, a parallel monitoring system has been implemented and partially tested.

1. GİRİŞ

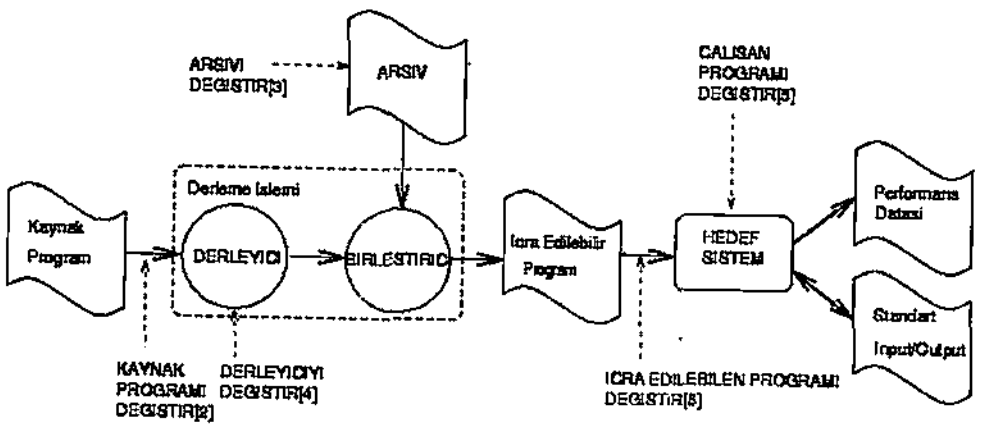
Bir programın performansını arttırabilmek için, o programın davranışının çalışırken gözlenebilmesi gerekir. Örneğin, sistem kaynaklarının ne şekilde kullanıldığı performans ayarı için gereklidir (performance tuning). Aynı şekilde, sistemin durumu ile ilgili bilgiler performans sorunlarının giderilmesi için önemlidir (performance debugging). Kısaca, program davranışını anlamak için gözlemleme bir ilk adımdır. Gözlemlemeyle "ne oldu?" sorusu cevaplanır, bu ise "neden oldu?" sorusuna cevap aramak için bir ön şarttır.

Bu çalışmada, yazılım tabanlı bir paralel gözlemleme sisteminin, gerçekleme öncesindeki mimari geliştirme çalışmaları sunulmustur. Mimari, gözlemleme sistemlerinin geniş kapsamlı bir analizinin ürünüdür. Özel bir sisteme bağlı olmadan geliştirilen bu mimari, paralel performans kavramlarının ve sorunlarının adreslenebileceği genel bir ortam oluşturmaktadır. Mimari, gerçeklemeyi hızlandıracak kapsamlı analiz ve doğruluk testine de imkan sağlamaktadır. Ayrıca, bir yazılım destekli gözlemleme sistemi, bu çalışmada geliştirilen mimari özellikler kullanılarak kısa zamanda gerçeklenmiştir ve kısmen test edilmiştir.

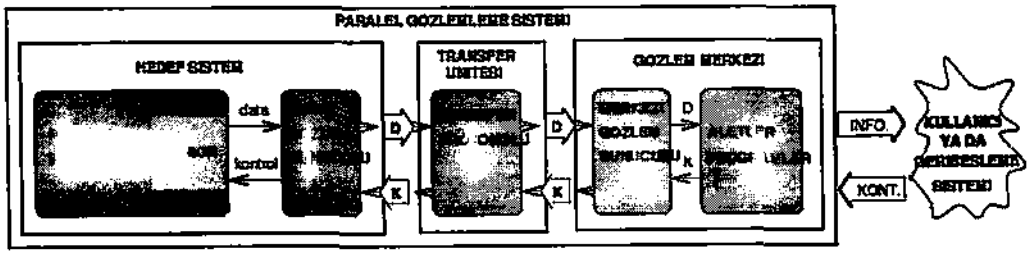
Bu çalışmanın kalan kısmı şöyle organize edilmiştir: Bölüm 2 yazılım yoluyla performans verisi toplama yöntemlerini kısaca tanımlar. Bölüm 3 mimarinin özelliklerini, Bölüm 4, 5 ve 6 sırasıyla hedef sistemi, transfer ünitesini ve gözlem merkezini açıklar. Bölüm 7 deney sonuçlarını sunar.

2. YAZILIM ENSTRUMANTASYONU

Performans verisi, çalışmakta olan bir programdan ancak enstrumantasyon yardımıyla elde edilebilir. Yazılım ağırlıklı gözlemleme sistemlerinde, yazılım parçalarından oluşan ilavelere sensör denilir [1]. Şekil-1'de görüldüğü gibi, sensörler, derleme işlemi öncesinde, sırasında, veya sonrasında hedef sisteme ilave edilirler [2, 3, 4, 5, 6].



Şekil-1 Derlemenin çesitli safhalarında enstrumantasyon



Şekil 2 Gözlemleme sisteminin fonksiyonel yapısı

Performans verisi ilgilenilen olay olduğunda sensör tarafından oluşturulur. Sensör uyarı politikası *olay-uyarıcılı* veya *zaman-uyarıcılı* olabilir (bunlar sırasıyle *izleme* ve *örnekleme* olarak bilinir) [7]. Olay-uyarıcılı politikada, ikincisinin aksine, performans verisi, olayın oluşumu ile eşzamanlı olarak üretilir. Her iki politikada da veri, üretildiğinde geçici bir bellekte *olay-kaydı* olarak saklanır. Genel olarak olay-kaydı, olayın olduğu zamanı belirten zaman-etiketi, olayın olduğu yeri bildiren yer-etiketi ve sistemin o anki durumuyla ilgili bazi ilave bilgileri içerir. Yazılım sensörleri orijinal sisteme sonradan ilave edildiğinden, sistemin karakterini değiştirecek etkiler yaratabilir, bu nedenle sensörler performans verisi toplanacak bir programa dikkatlice ilave edilmelidirler.

3. GÖZLEMLEME MİMARİSİ

Gözlemlenecek nesneye hedef sistem dersek, hedef sistemi minimum seviyede rahatsız edecek bir gözlemleme sistemi için, iyi tasarlanmiş alt birimlere ihtiyaç vardır. Örneğin, performans verisi üretmek üzere hedef sisteme ilave edilecek hızlı sensörlere, daha sonra bu veriyi toplayıp ana merkeze gönderecek etkin veri toplayıcılarına, ve bu verileri anında işleyip ekranda görüntü olarak sunacak programlara ihtiyaç vardır.

Şekil 1'de görüldüğü gibi mimari, üç ana bölümden oluşmaktadır: Hedef sistem, transfer ünitesi ve gözlem merkezi. Geliştirilen bu model seviyeler halinde de düşünülebilir. En tepede gözlemleme sistemi bütün giriş ve çıkış fonksiyonları ile görülmektedir. İkinci seviyede model üç zahiri bloğa bölünmüştür. Bu bloklar gerçekleme işlemini kolaylaştıracak şekilde, sistemi daha basit alt bölümlere ayırmaktadır. Aşağı seviyelere inildikçe ana bloklara ait daha fazla detaylar görülmektedir.

Gözlem merkezindeki Merkezi Gözlem Sunucusu (MGS) önce Yerel Gözlem Sunucu'larını (YGS) paralel sistemin tüm yerel düğümlerinde çalıştırır. Sonra, YGS'ler ilgili düğümlerde çalışması gereken hedef programları çalıştırırlar ve servise kabul ederler. YGS'ler ayrıca, veri transferi için hedef program ile kendisi arasında bir haberleşme kanalı (ya da veri transfer mekanizması) oluştururlar. Genelde haberleşme kanalları işletim sisteminin sunduğu soket arabirimi, boru (pipe) veya ortak-bellek (shared-memory) servisleri kullanılarak gerçeklenir. Buna ilaveten, YGS'ler ile MGS arasında çift yollu haberleşme kanalları olmalıdır. Böylece, performans verisi hedef sistemden MGS'ye doğru akarken, kontrol mesajları da ters istikamette hareket ederler. Hedef sistem ve gözlemleme sistemi aynı ağı kullandıklarından yarış gözlenir. Gözlemleme sistemi tarafından ağın kullanımı minimum seviyede tutulmalıdır.

4. HEDEF SİSTEM

Hedef sistem, performans incelemesi için gözlem altına alınan sistemdir. Hedef sistem, donanım ve yazılımdan oluşan bir bilgisayar sistemi olabileceği gibi, sadece yazılımdan oluşan bir program da olabilir. Bu çalışma için hedef sistem, *dağıtılmış paralel* programlardır. Gözlemleme sisteminin hedef sistemde gerçekleştirmesi gereken fonksiyonlar aşağıdaki gibidir:

- İlgilenilen performans verisinin üretilmesi.
- Performans verisinin geçici olarak bir bellekte saklanması (depolanması).
- Performans verisinin ön işlemden geçirilmesi (filtreleme, sıralama, zaman etiketinin ilave edilmesi vs.)
- Verilerin gözlem merkezine gönderilmesi.
- MGS'den kontrol mesajlarını kabul edip, veri toplama çalışmalarının ona göre yeniden düzenlenmesi.

Performans verilerinin toplanabilmesi için hedef sisteme donanım ya da yazılım sensörlerinin ilave edilmesi lazımdır. Yazılım sensörleri, bir başka yazılım sistemi yardımıyla, hedef programda önemli olayların olacağı noktalara yerleştirilir. Paralel performans için genelde, fonksiyona giriş-çıkış, mesaj gönderme-alma, bariyere ulaşma gibi olaylar önemlidir. Sensörler ilave edilmiş bir paralel program çalıştığında, ilgilenilen olay oluşumları YGS'ye rapor edilir. Paralel sistemin düğümlerindeki YGS'ler performans verilerini belirlenen politakaya göre toplarlar ve geçici bir bellek alanında saklarlar. Bu bellek alanı dolduğunda, performans verileri MGS'ye gönderilir. Performans verisinin yerel düğümlerde biriktirilerek merkeze gönderilmesinin amacı ağdaki trafiği azaltmaktır.

5. TRANSFER ÜNİTESİ

Transfer ünitesi, performans verilerinin ve kontrol mesajlarının, YGS ile MGS arasında taşınmasında kullanılan iletişim mekanizmasıdır (bkz. Şekil [1]).

Transfer ünitesinin donanım ve yazılımdan oluşan parçaları; ağ elemanları (ağ kartları, kablolar, köprüler, yönlendiriciler vs.) ve prosesler arası iletişim protokolleridir (TCP/IP, UDP/IP vs.). Ağ birimleri paylaşıldığından, gözlemlemeden dolayı ağdaki ilave yük, hedef programın icrasında gecikmelere yol açar.

Çok kullanılan yazılım destekli iletişim mekanizmalarından biri soket arabirimidir. Unix işletim sistemi, programcılara bu arabirimin hem bağlantılı ve hem de bağlantısız alternatiflerini gerçekleme imkanı sunar (sırasıyla TCP *"Transfer Control Protocol"* ve UDP *"User Datagram Protocol"*) [8]. Bu protokoller genelde IP protokolu (*"Internet Protocol"*) ile Internete mesaj göndermek-almak için kullanılır. TCP'de mesajların alıcı tarafından alınması garanti edilmiştir. Gönderici, karşı tarafın mesajı almasından sonra, alıcıdan "mesaj-alındı" (acknowledgment) işaretini geri almasıyla bir haberleşme dönemi sona erer. Gönderici proses, mesaj karşıya ulaşmadığı sürece, diğer işlerine devam edemez, bloke olur.

UDP'deyse mesajın alınıp alınmadığı kontrol edilmez. Eğer kontrol gerekiyorsa, programcı kendi yöntemini gerçekleyebilir. Mesaj gönderen proses, mesaj gönderme işlemi tamamlandığında, bloke olmadan, hemen bir sonraki komutun icrasına geçer, "mesaj-alındı" cevabını beklemez. Eğer prosesler arası haberleşmede, güvenilirlik çok önemli ise, herhangi bir ilave yapılmadan TCP kullanılabilir. Örneğin, PVM'de (*"Parallel Virtual Machine"*) ilgilenilen tüm olayların oldukları anda rapor edilmesinden emin olunmak için kullanıcı prosesleri ile yerel PVM sunucuları arasında TCP protokolu kullanılır [9]. Yerel ağlarda, UDP protokolu ile güvenilirlik daha hızlı ve doğrusal bir gecikme ile sağlanabilir. TCP ağlar arasında doğrusal bir gecikme sergileyemediğinden, birçok yerde (örneğin PVM'de) sunucular arası iletişim UDP protokolu ile sağlanır. PABLO ve AIMS gözlemleme sistemlerinde de veri transferi için soket arabirimi kullanılmıştır [10,2].

Uzak alt-program çağırma (Remote Procedure Call: RPC), Unix ortamında, prosesler arasında, yüksek seviyede müşteri-sunucu iletişim modelidir. Alt seviyelerdeki gerçekleme detayları; örneğin veri uyumu, soket arabirimi, sunucu gerçekleme gibi problemler, program üreteci derleyicisine bırakılmıştır. RPC, hedef prosesin adres alanının dışındaki programları çağırma imkanı sunduğundan, yerel bir program başka bir makinadaki programı çalıştırabilir, veri gönderip, sonuç alabilir. RPC, Paragon performans gözlemleme sisteminde, kontrol sinyallerinin yerel gözlemleme sunucularına gönderilmesinde kullanılmıştır [11]. Paragon'da, her düğümdeki YGS'ler, aslında birer RPC sunucularıdır.

Boru (pipe), bir makina sınırları içinde kullanılabilen bir diğer prosesler-arası iletişim mekanizmasıdır. Tek yönlüdür ve dosyalar gibi yazılıp okunabilirler. Soket arabiriminin aksine, borularda veri paketleme gerekli değildir. Gönderici proses bir ucundan yazarken, alıcı proses diğer ucundan okuma yapar. Boru, aynı makina sınırları içinde, veri transferinde soket arabiriminden daha hızlıdır.

Aynı makina sınırları içinde, prosesler-arası iletişim ortak-bellek kullanılarak da sağlanabilir. Bu metodla, aynı makinada prosesler-arası veri transferi, diğer bütün yöntemlerden daha hızlıdır. Ortak-bellek ile veri transferi Paradayn gözlemleme sisteminde kullanılmaktadır [5].

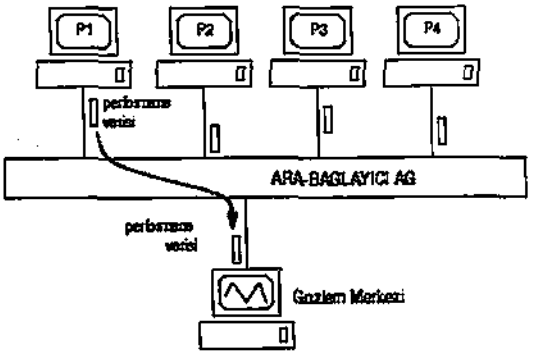
Son olarak, performans veri transferi için özel protokoller kullanılabilir. Örneğin, Hawlet Packard, VIZIR performans sisteminin esnekliğini arttırmak ve başka makinalara kolaylıkla uyaralanabilmesini sağlamak (portability) için, kullanıcı programları seviyesinde protokoller gerçeklemiştir [12].

6. GÖZLEM MERKEZİ

Gözlem merkezi, gözlem altındaki paralel sistemin tüm düğümlerinden performans verilerinin toplandığı merkezi bir istasyondur(bkz. Şekil [3]). Bu merkezde, yerel düğümlerdeki hedef prosesleri çalıştırabilecek, durdurabilecek kabiliyette bir Merkezi Gözlem Sunucusu (MGS) bulunur. MGS dinamik performans verilerini YGS'lerden toplar. Buradaki veri işleyen araç programlar, çalışmakta olan sistemin görüntüsünü oluşturur. Gözlem merkezinin yaptığı işler söyledir:

- Hedef sistemin ve gözlemleme olaylarının kontrolu için bir konsol sağlanması.
- Gelen verilerin geçici olarak saklanması.
- Verilerin ön işlenmesi (olayların tutarlı bir şekilde sıralanması, verilerin diğer programların anlayabileceği genel bir formata konması).
- Verilerin bir kopyasının (animasyon için) kalıcı belleklerde saklanması.
- Verilerin ilgili araç programlara dağıtılması.

Araç-programlar, performans verilerini işlerler, sonuçları tablo ya da grafik formatında ekrana yansıtırlar. Bu programlar, on-line veya off-line olmak üzere iki ana gruba ayrılır. On-line araç-programları veriyi üretildiği anda işlerler. Off-line araç-programları ise, daha önce saklanmış performans verilerini kullanarak, paralel daha önce çalışmış sistemin animasyonununu, arzu edilen metrikte yaparlar. Eger performans verilerini saklamak sorun olursa, (genellikle paralel programlar saklanamayacak kadar çok performans verisi üretirler) off-line araçlar kullanılamazlar. Bununla beraber, off-line araçlar genelde hedef programın çalışmasını daha az rahatsız ederler.



Şekil-3 Performans verisinin Gözlem Merkezi'nde toplanması

Bir *sistemin farklı özelliklerini ölçebilmek için*, gözlemleme merkezi, veriyi farklı şekillerde yorumlayabilen araç-programlar sağlamalıdır. Grafiksel çıktıların anlaşılması genelde tablolara göre daha kolaydır. Fakat, bu tür çıktılar için daha fazla sistem kaynakları (CPU ve bellek) gereklidir. Bu nedenle, on-line paralel performans görüntülemeleri için genelde ayrı bir iş-istasyonu kullanılır, ve böylece hedef sistemin kaynakları daha az kullanılmış olur.

Paralel performans sistemlerinde gözlem merkezinde halledilmesi gereken bir diğer problem de, performans verisinin zamana göre olayların oluş sırası ile tutarlı bir şekilde sıralanmasıdır. Yani, herbir olaya global olarak tutarlı olan bir zaman etiketi iliştirilmeli, ya da başka çözümler bulunmalıdır. Dağıtılmış paralel sistemlerde iş-istasyonları arasında saat eşzamanlaması sağlamak amacıyla "Network Time Protocol" (NTP) geliştirilmiştir (RFC 1119 da tanımlanmıştır). Fakat, paralel sistemlerde olay *oluşumu çok hızlı olduğundan, bu eşzamanlama* metodu olayların sıralanmasında yeterli tutarlılık sağlayamamaktadır. Bu duruma çözüm olarak Lamport, *lojik-saatler* kavramını geliştirmiştir [13]. Bu kavramda, bir prosesin icrası, ardışıl olaylar ile karakterize edilmiştir. Lojik saatler gözlemleme sistemlerinde eşzamanlılık problemlerini gidermek için yaygın olarak kullanılmaktadır. Program icrasının ardından, olayların nedensel olarak sıralı olup olmadığı kontrol edilir, değilse sıralama düzeltilir.

7. GERÇEKLEME VE DENEY SONUÇLARI

Bu mimari *kullanılarak yazılım tabanlı bir* gözlemleme sistemi gerçeklendi. Gerçekleme ve test işlemleri paralel sistemin düğümlerinde tamamlandı. Gözlem merkezi ile transfer ünitesi gerçeklendi fakat karşılaştırmalı testleri henüz tamamlanmadı. Gerçeklenen ve testleri tamamlanan kısımların sonuçları bu bölümde sunulmuştur.

Gerçeklenen gözlemleme sisteminin özellikleri aşağıdaki gibidir:

- Binary sensörler geliştirilen bir program yardımıyla, icra edilebilir programlara *(executable) ilave edildi*
- Hibrid (örnekleme ve izlemeden oluşan karma) veri toplama politikası kullanıldı.
- Yerel düğümlerde veri toplamak için ortak-bellek iletişim kanalı olarak kullanıldı.
- YGS'ler ile MGS arasında TCP/IP kullanıldı.
- Performans verileri, merkezi sistemde diske yüklendi ve sonuçlar off-line gözlemleme araçları ile tablolardan incelendi.

Enstrümantasyon sisteminin, ve yerel düğümlerde gerçeklenen gözlemleme sisteminin performansını karşılaştırabilmek için, SPLASH-2 benchmark *suitinden* RADIX, FFT, ve LU programları kullanıldı. Tablo-1 kaç tane alt-programa sensör ilave edildiğini, ne kadar performans verisi toplandığını ve örnekleme hızını göstermektedir. Deneyler 10 defa tekrar edildi ve standart sapmalar hesaplanarak tabloya ilave edildi.

Yeni sistem, Sun-HyperSparc-20, Solaris 5.6 Unix iş istasyonlarında denendi ve sonuçlar *yerel düğümler* için gprof ile karşılaştırıldı. İş istasyonlarında 64 Mbyte ana bellek vardı ve 60 MHz saat hızıyla çalışmaktaydı. Deneyler sırasında duvar-saati (wall-clock) kullanıldı ve sistemde sadece test programı çalışıyordu.

Gprof uniprosesor sistemler için GNU tarafından geliştirilmiş bir gözlemeleme sistemidir. C ve Fortran programlarınıyla kullanılabilen gprof, sonuçları off-line olarak tablo formatında sunar. Gprof performans verisini örnekleme politikasına göre toplar[4].

İlk olarak test programımlarının enstrumantasyonla ne kadar genişlediğine baktık. Tablo 2'de orijinal ve sensörler ilave edilmiş programların boyutlarının ne kadar büyüdüğü görülmektedir. Gerçeklenen sistem gprof'a göre programları yaklaşık %60 daha az büyüttü. Bunun nedeni ise gprof YGS'leri sensörlerle birlikte programa ilave etmekteydi.

Sonra, bir orijinal haliyle ve bir de enstrument edilmiş haliyle her iki gözlemleme sistemleri için programları çalıştırdık ve çalışma sürelerini ölçtük. İcra süreleriyle ilgili sonuçlar Tablo-3'te sunulmuştur. Gerçeklenen gözlemleme sistemi yerel düğümlerde en az gprof kadar, ve daha iyi sonuçlar vermiştir.

8. SONUÇ

Bu çalışmada, yazılım destekli bir paralel gözlemleme sistem mimarisi sunuldu. Mimari var olan sistemlerin kapsamlı bir analizi sonucunda geliştirildi. Genel olmasına dikkat edilen, ve kavramların yerleşmesinde *kullanılmak üzere geliştirilen mimari, daha sonra bir* gözlemleme sisteminin gerçeklemesinde kullanıldı. Kısmen tamamlanan sistem ile yapılan deney sonuçları gprof ile karşılaştırıldı. Sonuçlar sistemin tamamını gerçeklemek için ümit vericidir

Table 1. Enstrumantasyon özeti

Program adı	Sensorlü alt-program sayısı	Toplanan data miktarı (Byte)	Örnekleme hızı (s.)
RADIX	26	221	0.1
FFT	38	1588	0.1
LU	35	222	0.1

Table 2. Gerçeklenen sistem programları yaklaşık 4.6. gprof ise 10 defadan fazla büyüttüyor.

Program adı	Orijinal boyut(Byte)	Bizim enstrumantasyon ile(Byte)	Gprof ile (Byte)
RADIX	32,997	160,197	346,040
FFT	42,794	160,871	371,374
LU	29,741	160,303	371,742

Tablo 3. Orijinal ve enstrüment edilmiş icra süreleri (gprof ile karşılaştırmalı olarak)

Program adı	Orij. süre(s.)(stdev)	Bizim enstrüman. ile(s.)(stdev)	Gprof ile (s.)(stdev)
RADIX	14.27(0.04)	16.48(0.06)	18.78(0.06)
FFT	61.08(9.35)	91.90(4.84)	89.50(7.82)
LU	21.49(0.08)	22.30(0.008)	29.43(1.11)

KAYNAKLAR

[1] R. Snodgrass, A Relational Approach to Monitoring Complex Systems, ACM TRANSACTIONS ON COMPUTER SYSTEMS, vol: 6, pp: 157-196, May 1988.

[2] Jerry Yan, Performance Tuning with AIMS-An Automated Instrumentation and Monitoring System for Multicomputers, PROCEEDINGS OF THE 27TH HICS, WAILEA, HAWAII, January 1994.

[3] B. P. Miller, M. Clark, J. Hollingsworth, S. Kierstead, S. Lim, and T. Torzewski, IPS2: The Second Generation of a Parallel Program Measurement System, IEEE TRANSACTİONS ON PARALLEL AND DİSTRİBUTED SYSTEMS, vol:1(2), pp: 206-217, April 1990.

[4] Susan L. Graham, Peter B. Kessler, and Marshall K. McKusick, Gprof: A Call Graph Execution Profiler. PROCEEDINGS, SIGPLAN '82 SYMPOSIUM ON COMPILER CONSTRUCTION, pages 120-126, June 1982.

[5] Jeffrey K. Hollingsworth, Barton P. Miller, and Mark D. Callaghan, The Paradyn Parallel Performance Tools and PVM, SIAM PRESS, 1994.

[6] James Larus and Thomas Ball, Rewriting Executable Files to Measure Program Behavior, SOFTWARE: PRACTICE AND EXPERIENCE, vol: 24, pp: 197-218, February 1994.

[7] R. Hoffmann, R. Klar, B. Mohr, A. Quick, and M. Siegle, Distributed Performance Methods, Tools, and Applications, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, pp: 585-597, June 1994.

[8] D. E. Comer, Internetworking with TCP/IP, Vol III: Client-Server, Programming and Applications, PRENTICE-HALL, 1993.

[9] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam, PVM 3 User's Guide and Reference Manual, OAK RIDGE NATIONAL LABS, September 1994.

[10] V. S. Adve, J. M. Crummey, M. Anderson, K. Kennedy, J. C. Wang, and D. A. Reed, Integrated Compilation and Performance Analysis Environment for Parallel Programs., PROCEEDINGS OF SUPERCOMPUTING '95, December 1995.

[11] B. Ries, R. Anderson, W. Auld, D. Breazel, K. Callagham, E. Richards, and W. Smith, The Paragon Performance Monitoring Enviroment, PROCEEDINGS OF SUPERCOMPUTING '93, pp: 850-859, Portland, November 1993.

[12] H. Ming C., A. H. Karp, A. Waheed, and M. Jazayeri., VIZIR: An Integrated Environment for Distributed Program Visualization,.PROCEEDINGS OF WORKSHOP ON MODELING, ANALYSIS AND SIMULATION OF COMPUTER AND TELECOMMUNICATION SYSTEMS, (MASCOTS '95) Tools Fair, January, 1995.

[13] Leslie Lamport, Time, Clocks, and the Ordering of Events in a Distributed System, COMMUNICATIONS OF THE ACM, pp: 558-565, July 1978.