Journal of Korea Multimedia Society Vol. 13, No. 11, November 2010(pp. 1594-1609)

예측 가능한 입출력 대역폭을 제공하는 서비스 기반의 디스크 입출력 제어

강동재†, 이평화††, 정성인†††

요 약

최근 가상화와 클라우드 컴퓨팅 이슈가 대두되면서 서비스 품질과 성능 안정성을 보장하기 위한 대안으로서 시스템 자원 관리의 필요성이 부각되고 있으며, 디스크 입출력 자원은 오랜 기간 동안 심각한 성능 병목으로 인식되어 왔다. 다수의 서비스들이 입출력 자원에 대하여 경쟁 상태에 있는 경우, 낮은 중요도를 갖는 서비스 또는 프로세스들에 의하여 제한된 입출력 자원들이 점유되는 경우가 빈번하게 발생하며, 이는 중요 서비스의 품질 및 성능을 저하시키는 요인으로 작용한다. 또한, 특정 프로세스 또는 서비스가 사용할 수 있는 입출력 자원을 예측할 수 없는 경우에는 서비스의 성능 안정성 및 품질을 보장할 수 없으며 제한된 자원에 대한 효율적인 사용을 어렵게 만드는 문제점을 갖는다. 따라서, 본 논문에서는 상기 문제점의 개선을 위하여 예측 가능한 입출력 대역폭을 제공하는 서비스 기반 디스크 입출력 제어 방식을 제안하며, 제안 방식은 서비스 기반의 예측 가능한 입출력 대역폭을 제공함으로써, 안정적인 서비스 품질 및 성능을 유지하도록 하며, 제한된 입출력 자원의 효율적인 사용을 가능하게 한다.

Service based Disk I/O Control supporting Predictable I/O Bandwidth

Dong Jae Kang†, Pyoung Hwa Lee††, Sung In Jung†††

ABSTRACT

In the case that multiple services are in race condition for limited I/O resource, the services or processes with lower priority occasionally occupy most of limited I/O resource. And it decreases QoS and performance of important services and makes it difficult to efficiently use limited I/O resource. Although system administrator allocates I/O resource according to priority of process, he/she can't know or expect how much resource will be used by the specific process. Due to these reasons, it causes the problem that he/she can't guarantee the service QoS and performance stability. Therefore, in this paper, we propose service based disk I/O control supporting predictable I/O bandwidth to resolve upper problems. Proposed I/O control guarantees the service QoS and performance stability by supporting the service based predictable I/O bandwidth and it makes limited I/O resource to be efficiently used in respect of service.

Key words: Service, predictable I/O Bandwidth, Range Share Technique(구간 제어 기법), Proportion Share Technique(비율적 제어 기법)

※ 교신저자(Corresponding Author) : 강동재, 주소 : 대전시 유성구 가정로 138(305-700), 전화 : 042)860-1561, FAX : 042)860-6699, E-mail : djkang@etri.re.kr
접수일 : 2010년 5월 23일, 수정일 : 2010년 8월 9일
완료일 : 2010년 9월 20일

† 정회원, 한국전자통신연구원 선임연구원
(E-mail : djkang@etri.re.kr)
†† 정회원, 한양대학교 전자컴퓨터통신공학과 석사과정
(E-mail : phlee@rtcc.hanyang.ac.kr)
††† 정회원, 한국전자통신연구원 책임연구원
(E-mail : sijung@etri.re.kr)

1. 서 론

최근 서비스 품질과 성능 안정성을 위한 시스템 자원 관리의 필요성이 부각되고 있으며 자원 사용량에 따른 과금을 기반으로 하는 유틸리티 컴퓨팅 환경에 있어서도 예측 가능한 시스템 자원의 제공은 주요한 이슈로서 거론되고 있다[1,2]. 또한, 새로운 컴퓨팅 인프라와 서비스 개념들이 꾸준한 진화를 거듭하면서 서비스를 구성하는 요소들의 복잡성도 함께 증가하였으며, 그로 인한 서비스의 품질 및 성능 안정성의 유지는 더욱 어려워지고 있다[3].이러한 시스템 자원 관리 대상 중에서, 디스크 입출력 자원은 오랜 기간 동안 심각한 성능 병목으로 인식되어 왔으며 낮은 성능으로 인하여 CPU 및 메모리 자원의 효율적인 사용을 어렵게 하고 있다. 시스템 자원의 관리는 대부분의 운영체제에서 사용되는 프로세스 기반 자원 관리[4]와 서버 가상화 환경에서 사용되는 시스템 기반의 자원 관리가 범용적으로 사용되고 있다[5,6]. 하지만, 특정 서비스의 품질 및 성능 안정성을 효과적으로 보장하기 위해서는 프로세스 또는 시스템 단위가 아닌 서비스 단위 자원 관리 방식이 요구되며, 이는 서비스를 구성하는 모든 프로세스들의 자원 요구사항이 동시에 만족되어야 함을 의미한다.

실제 서비스 제공자 측면에서 자원 관리가 요구되는 시점은 예측이 어려운 과도한 입출력이 발생하는 시점이나 시스템 상에서 구동하는 여러 프로세스 또는 서비스들이 동시에 과도한 자원을 요청하는 시점이다. 이러한 환경에서 각 서비스는 서비스 제공자 측면에서 차별적인 중요도 또는 우선순위를 갖는다. 하지만 중요도가 높은 서비스가 상대적으로 많은 입출력 자원을 점유하도록 보장하기 어렵기 때문에 낮은 중요도를 갖는 서비스 또는 프로세스들에 의하여 제한된 입출력 자원이 점유되는 경우가 빈번하게 발생한다[3]. 이는 중요 서비스의 품질 및 성능을 저하시킬 수 있는 요인으로 작용하며 제한된 자원의 효율적인 사용을 저해하는 문제점을 야기한다. 또한, 우선순위에 따라서 디스크 입출력 자원을 할당하더라도 특정 프로세스 또는 서비스가 사용할 수 있는 입출력 자원을 예측할 수 없기 때문에 서비스의 성능 안정성 및 품질을 보장하기 어렵다는 문제를 갖는다.

따라서, 본 논문에서는 앞서 기술한 문제점들의 개선을 위하여 예측 가능한 입출력 대역폭을 제공하는 서비스 기반 디스크 입출력 제어 방식을 제안한다. 제안하는 방식은 서비스 기반의 디스크 입출력 자원 관리를 위하여 프로세스 그룹을 하나의 관리 객체로 제공하는 서비스 관리기를 포함하고 서비스의 중요도에 기반한 예측 가능한 대역폭을 제공하는 입출력 제어 기법을 제공한다.

본 논문에서 제안하는 입출력 제어 방식은 성능 평가에서 제시하는 바와 같이, 서비스 기반의 예측 가능한 입출력 대역폭을 제공함으로써, 안정적인 서비스 품질 및 성능을 유지하는데 효과적이며, 제한된 입출력 자원의 효율적인 사용을 가능하게 한다.

2. 관련 연구

2.1 프로세스 그룹의 관리

본 절에서는 상기와 같은 다수의 프로세스들을 하나의 객체로서 관리하기 위한 대표적인 기존 연구들에 대하여 살펴보도록 한다. 다수의 프로세스들을 하나의 객체로 관리하기 위한 프로세스 그룹에 대한 연구는 프로세스 그룹 기반의 통합적인 자원 관리와 속성 및 목적이 다른 프로세스들간의 네임스페이스 분할(Name Space Isolation) 이라는 두 가지 방향으로 꾸준히 수행되어 왔다[7,8].

시스템 자원의 통합 관리에서는 관련된 프로세스들이 사용하는 자원의 양을 전체적으로 파악하거나 특정 그룹의 프로세스들이 사용할 수 있는 자원을 관리하기 위한 목적으로서 연구되었으며, 네임스페이스 분할에서는 프로세스 그룹들의 실행 환경을 서로 격리하여 각각의 프로세스 그룹들이 접근할 수 있는 자원 및 환경을 독립적으로 운영하기 위한 목적으로서 연구되었다. 네임 스페이스 분할을 위한 연구는 현재 범용적으로 사용되고 있는 가상화 환경의 기반 기술로 사용되고 있다[5,6]. 선행 연구의 대부분은 소스가 공개된 Unix 계열의 운영체제에서 구현되었으며, 표 1은 프로세스 그룹의 관리를 위한 선행 연구들의 특징을 정리한 표이다.

2.2 프로세스 그룹 기반의 입출력 제어

본 절에서는 기존의 프로세스 그룹 기반의 입출력 자원 관리와 관련한 선행 연구들에 대하여 살펴보도록 한다. CPU 및 메모리 자원 제어기와 비교하여,

표 1. 프로세스 그룹 관리를 위한 기존 연구들

선행 연구	특 징
CKRM [8]	프로세스 그룹을 Class라는 개념으로 정의 프로세스 그룹 관리를 위한 대표적인 연구 구현의 복잡성으로 인한 낮은 활용성 타 제어기를 위한 확장성의 부족 CPU, 메모리를 중심으로 한 자원관리 부분을 구현
PAGG [9]	프로세스 그룹을 Process Aggregate라는 개념으로 정의 프로세스 그룹화를 위한 기본 기능만을 구현한 단순한 구조 Job Container라는 향상된 프로세스 관리 모듈의 기반으로 사용 Comprehensive System Accounting(CSA) 프로젝트에 적용
Cgroup [7]	프로세스 그룹을 Control Group이라는 개념으로 정의 프로세스 그룹 기반의 다양한 모듈의 추가를 고려한 확장성 지원 CPU, 메모리, 입출력, 자원 모니터링 등 많은 제어 모듈들의 개발을 진행 범용 인터페이스의 지원으로 사용 편의성 제공 계층적 프로세스 그룹 모델 제공

프로세스 그룹 기반의 입출력 제어를 위한 기술들은 대부분의 범용 운영체제에 반영되지 않고 있으며, 현재 공개 소프트웨어 커뮤니티 등을 통한 소수의 연구 개발이 진행 중이다[10-12]. 표 2는 리눅스 운영체제 환경을 기반으로 개발중인 프로세스 그룹 기반의 입출력 제어기에 대한 비교표이며 제안하는 입출력 제어 방식을 포함한다.

입출력 제어기 수준의 구현은 기존 입출력 제어기를 대체하는 형태이므로 제어기에 의존적이며, 블록 레이어 아래에 존재하므로 블록 수준에서 제공하는 소프트웨어 레이드 등에 의한 가상 입출력 디바이스에 대한 제어가 불가능하다는 단점을 갖는다. 반면, 블록 수준의 구현은 시스템 입출력 제어기에 상관없이 상위 수준에서 입출력 제어를 수행할 수 있으며, 블록 수준에서 제공하는 가상 입출력 디바이스에 대한 제어가 가능하다. 또한, 대부분의 제어기들은 입출력 제어 정확도의 감소 문제를 발생시키는 지연 입출력의 처리 기능을 제공하고 있지 않다. 지원하는 입출력 제어 기법의 측면에서 보면, 2 Layer CFQ와 2 Level CFQ는 비율적인 입출력 제어 기법을 지원하고 있으나, 입출력 대역폭의 할당 비율에 따른 제어 정확도가 낮고 예측 가능한 자원의 분배 방식을 지원하지 않는다. 또한, 입출력 주체 중심의 제어 방식이기 때문에 각 입출력 주체는 시스템에 존재하는 모든 디바이스에 대하여 동일한 대역폭의 할당 값을 갖는다는 한계성이 존재한다.

Io-throttle 제어기는 입출력 대역폭에 대한 상한 제한 기법(Limitation Share Technique)을 지원하며, 가용한 최대 대역폭을 제한하는 방식이기 때문에 일정 수준의 예측 가능한 자원의 분배 방식이라고 볼 수 있다. 또한, 입출력 디바이스 중심의 제어 방식이므로 동일한 입출력 주체에 대하여 입출력 디바이

표 2. 프로세스 그룹 기반 입출력 제어기의 특성 비교

	2 Layer CFQ[10]	2 Level CFQ[11]	Io-throttle[12]	제안하는 입출력 제어 방식
구현 수준	System Scheduler Layer		General Block Layer	General Block Layer
지연 입출력 관리	Not support		Not support	Support
입출력 제어 기법	Proportion Share		Limiting Share	Range Share, Proportion Share
입출력 대역폭의 예측 가능성	Not predictable		predictable	predictable
제어 기준	I/O Owner		I/O Device	I/O Device
잔여 자원의 관리 방식	work-conserving		Non work-conserving	work-conserving

스 마다 차별화된 대역폭의 할당이 가능하다. 표 2에서, 잔여 자원의 관리 방식은 할당된 대역폭의 일부가 사용되지 않을 때의 처리 방식을 의미하며, work-conserving 방식은 사용하지 않는 입출력 대역폭을 다른 입출력 주체들이 사용할 수 있도록 허용하는 방식이며, Non work-conserving은 잔여 입출력 대역폭이 존재하더라도 다른 입출력 주체에게 사용을 허용하지 않는 방식이다.

제안하는 입출력 제어 방식은 예측 가능한 입출력 대역폭을 제공하는 구간 제어 기법(Range Share Technique) 및 비율적 제어 기법(Proportion Share Technique)을 지원하며, 입출력 디바이스 중심의 제어 방식을 사용하므로 입출력 주체는 각 입출력 디바이스에 따라 차별적인 대역폭의 할당 값을 갖는다.

3. 예측 가능한 대역폭을 제공하는 서비스 기반의 입출력 제어

본 장에서는 예측 가능한 디스크 입출력 대역폭을 제공하기 위한 서비스 기반 디스크 입출력 제어에 대하여 기술하며 제안하는 방식은 서비스 관리기와 입출력 제어 기법으로 구성된다. 본 논문에서, 제어 디바이스는 입출력 대역폭을 제공하는 디바이스들 중에서 입출력 대역폭의 공유를 위하여 등록된 디바이스를 의미하며, 제어 대상이란 제어 디바이스의 입출력 대역폭을 공유하기 위하여 제안하는 입출력 제어 방법에 등록된 프로세스 그룹인 개별 서비스를 의미한다.

3.1 서비스 관리기

서비스 관리기는 서비스 기반 디스크 입출력 제어를 위하여, 프로세스 그룹을 하나의 관리 객체로 제공하며, 단일 업무와 관련된 다수의 프로세스들을 하나의 서비스로 관리함으로써, 서비스를 구성하는 프로세스들에 대한 일관된 자원 관리의 기반을 제공한다.

그림 1은 서비스 관리기의 전체 구조를 설명한 그림으로 서비스 관리 인터페이스 부분과 서비스의 관리를 위한 핵심 기능 부분으로 구분할 수 있다. 서비스 관리 인터페이스는 서비스의 생성 및 삭제, 서비스의 관리를 위한 상위 인터페이스를 제공하는 부분이고 서비스의 관리 부분은 생성된 서비스에 포함될 프로세스의 추가, 기존 프로세스의 삭제 및 서비스 간의 프로세스 이동 등의 기능을 제공한다.

시스템에 존재하는 모든 프로세스들은 서비스 관리기의 구동 시, 루트 서비스의 프로세스 리스트 상에서 관리되기 때문에 루트 서비스의 식별자를 참조하게 된다. 그림 1에서 서비스 1, 서비스 2, 서비스 3은 입출력의 제어를 위하여 생성된 서비스들이며, 생성된 서비스들은 루트 서비스를 시작점으로 하는 서비스 리스트의 객체로서 관리된다. 서비스의 생성 시에는 고유한 식별자가 할당되며 포함되는 프로세스들은 서비스에 대한 프로세스 추가를 통하여 해당 서비스의 프로세스 리스트에 등록된다. 새로운 서비스에 등록된 프로세스는 루트 서비스에서 관리되던 프로세스이며, 새로운 서비스에 대한 등록 연산이 발생하면서, 루트 서비스의 프로세스 리스트로부터 삭제된다. 기존 서비스로부터 특정 프로세스의 삭제는 해당 서비스의 프로세스 리스트로부터 삭제를 의미하며, 루트 서비스의 프로세스 리스트에 재등록된다. 생성된 서비스들 사이에서도 프로세스의 이동이 발생할 수 있으며, 서비스 1에 존재하는 특정 프로세스를 서비스 3으로 이동하는 연산으로 정의할 수 있다. 프로세스의 이동 연산은 루트 서비스가 관여하지 않으며 생성된 서비스들 사이에 프로세스의 재구성 연산 등에서 요구되는 기능이다. 앞에서 기술한 바와 같이, 프로세스들은 언제나 특정 서비스에 포함되어 있는 상태이며, 동시에 두 개 이상의 서비스에 등록되는 일은 발생하지 않는다.

그림 1에서와 같이, 각 서비스에서는 해당 서비스에 포함되는 프로세스들의 리스트를 관리하며, 서비스에 포함된 각 프로세스들은 자신이 포함된 서비스의 정보를 직접 참조하기 위한 수단을 갖는다. 이는 프로세스가 포함된 서비스 식별자의 참조 시에 전체 프로세스 리스트를 검색하는 연산을 막기 위한 것이며, 개별 프로세스로부터 직접 서비스 식별자를 참조함으로써 연산의 수행 속도를 개선하기 위함이다.

그림 2에서와 같이, 임의의 서비스에 포함된 프로세스가 입출력을 수행하는 경우, 입출력 제어 부분은 해당 프로세스가 요청한 입출력을 처리하게 된다. 이 때, 입출력 제어 부분에서는 해당 입출력 요청을 발생시킨 프로세스 정보를 참조하게 되며, 프로세스 정보로부터 해당 프로세스가 포함된 서비스의 식별자를 참조하게 된다. 따라서, 입출력 제어 부분은 입출력 요청을 발생시킨 프로세스 기반이 아닌, 프로세스들

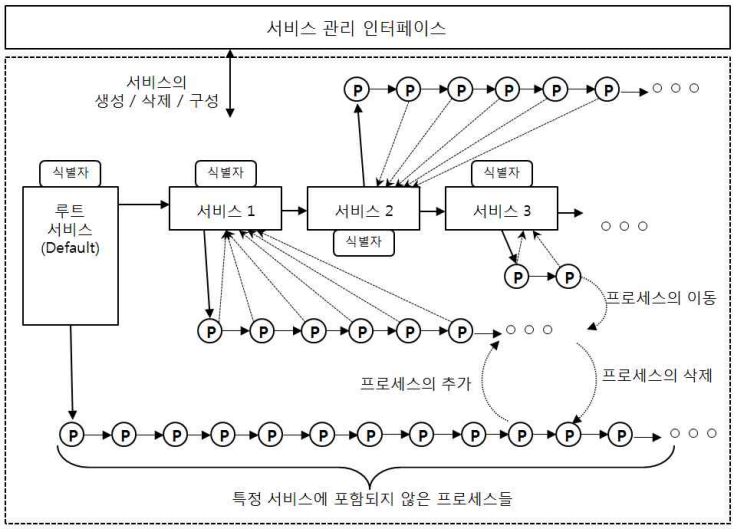

그림 1. 서비스 관리기의 구조

을 포함하고 있는 서비스 기반의 입출력 제어를 수행하게 된다. 즉, 서비스 관리자는 입출력을 수행하는 프로세스가 포함된 서비스 식별자를 입출력 제어 부분에 제공하며, 입출력 제어 부분은 서비스 식별자를 이용하여 서비스 기반의 입출력 제어를 수행한다.

또한, 제안하는 서비스 관리기에서는 *fork*, *exit*과 같은 시스템 시그널에 의한 프로세스의 생성 및 종료는 해당 시그널의 발생 시점에 처리될 수 있도록 기존 인터페이스의 구현부분을 수정함으로써 자동화한다. 이러한 기존 인터페이스의 수정은 임의의 서비스에 포함된 프로세스 A가 시스템의 *fork* 시그널에 의하여 자식 프로세스 A'를 생성하는 경우에 A'를 부모 프로세스 A가 속한 서비스에 자동으로 추가하며, 시스템의 *exit* 시그널에 의하여 종료되는 프로세스 B를 해당 서비스로부터 자동 삭제한다. 따라서, 관리자는 의도하는 정책에 의한 서비스의 추가, 삭제, 이동만을 수행하게 되며, 개별 프로세스의 수행 중에 시스템 시그널에 의하여 발생하는 프로세스의 생성 및 종료는 자동 반영된다는 장점을 제공한다.

3.2 예측 가능한 입출력 대역폭을 제공하는 입출력 제어 기법

본 절에서는 제안하는 입출력 제어 기법에 대하여 기술하며, 구간 제어 기법과 비율적 제어 기법으로 구분한다. 입출력 제어 기법은 제어 디바이스의 입출력 대역폭을 제어 대상들 간에 특정 기준에 의하여

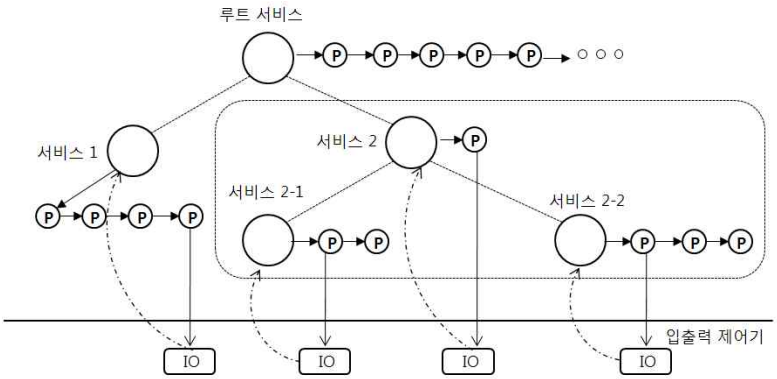

그림 2. 서비스 관리기와 입출력 제어기의 연동

공유하기 위한 입출력 관리 방식을 의미하며, 하나의 물리적 또는 논리적 디스크상에 존재하는 모든 제어 대상에 대하여 동일한 입출력 제어 기법이 적용된다. 입출력 대역폭의 제어에 있어서, 제어 대상에 따른 차별적 대역폭을 제공하기 위해서는 입출력 권한의 부여를 위한 제어 대상의 선정 알고리즘과 부여된 입출력 권한의 지속 시간 및 입출력 허용 양을 산정하기 위한 제어 수단의 적절한 운영이 요구되며 제공하고자 하는 입출력 제어 기법에 따라 서로 다른 운영 방식이 적용된다.

3.2.1 정량적 대역폭을 제공하는 구간 제어 기법 (Range Share Technique)

구간 제어 기법은 최소 대역폭 및 최대 대역폭을 상하 경계로 가지며 최소 대역폭 보다는 크고, 최대 대역폭 보다는 작은 정량적인 구간 대역폭을 제공하기 위한 입출력 제어 기법이다. 따라서, 구간 제어 기법은 일정량의 대역폭이 항상 보장되어야 하는 IPTV나 동영상 서비스와 같은 응용에 적합한 기법이며 정량적인 입출력 대역폭을 제공하기 위하여 입출력 모드(I/O Mode)라고 하는 새로운 제어 수단을 도입하여 입출력 제어를 수행한다.

입출력 모드는 제어 대상의 입출력 양을 의미하는 입출력 토큰 값을 기반으로 운영되며, 전역 입출력 모드(Global I/O Mode)와 지역 입출력 모드(Local I/O Mode)로 구분한다. 전역 입출력 모드는 제어 대상의 선정 방식을 결정하기 위한 것이며, 지역 입출력 모드는 제어 대상에게 할당되는 타임 슬라이스의 크기를 결정하기 위한 것이다.

제안하는 구간 제어 기법은 할당된 대역폭을 전역 타임 슬라이스(1초)마다 만족시키는 방식으로 각 전역 타임 슬라이스 내에서 개별 제어대상에게 할당되는 타임 슬라이스는 지역 타임 슬라이스라고 한다.

(가) 지역 입출력 모드(Local I/O Mode)

지역 입출력 모드는 특정 제어 대상이 수행한 입출력 양에 따라 변화하는 제어 대상의 속성으로 제어 대상의 입출력 긴급도에 따라서 차별적인 타임 슬라이스를 할당하기 위한 제어 수단이며 아래와 같이, 서로 다른 네 개의 입출력 모드 상태로 구분한다.

- *L_DEFAULT_IO* : 최소 및 최대 대역폭으로 정의되는 구간 대역폭이 할당되지 않은 제어 대상의 입출력 모드 상태($G_A.min$ and $G_A.max == NULL$)
- *L_MIN_BW_IO* : 수행한 입출력의 양이 반드시 보장되어야 하는 최소 대역폭을 만족하지 않은 제어 대상의 입출력 모드 상태 ($G_A.io < G_A.min$)
- *L_RANGE_IO* : 수행한 입출력의 양이 최소 대역폭 보다 같거나 많고 최대 대역폭 보다 적은 제어 대상의 입출력 모드 상태 ($G_A.min =< G_A.io < G_A.max$)
- *L_NO_IO* : 전역 타임 슬라이스 내에서 수행한 입출력의 양이 최대 대역폭 보다 같거나 많은 제어 대상의 입출력 모드 상태 ($G_A.max =< G_A.io$)

제어 대상 A를 G_A라고 하고, G_A의 현재 입출력 대역폭을 $G_A.io$, G_A의 최소 대역폭을 $G_A.min$, 최대 대역폭은 $G_A.max$라고 정의한다. $G_A.io$의 값은 전역 타임 슬라이스 내에서 G_A가 입출력을 수행함에 따라서 사용된 토큰의 값으로 산정되며, $G_A.min$ 및 $G_A.max$의 값은 제어 대상에 대한 구간 대역폭의 할당 시, 최소 및 최대 대역폭의 값에 해당하는 토큰의 값으로 산정된다.

그림 3은 G_A의 입출력이 진행됨에 따라서 변화되

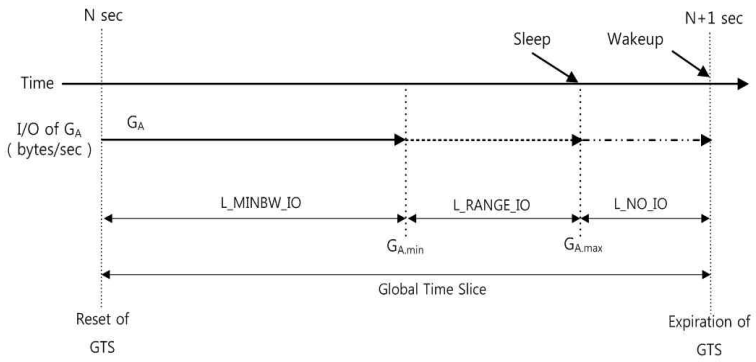

그림 3. 지역 입출력 모드의 운영 방식

는 G_A의 지역 입출력 모드의 상태를 나타낸 그림이며, 지역 입출력 모드의 상태는 *L_MINBW_IO, L_RANGE_IO, L_NO_IO*의 순으로 변화한다. 구간 제어 기법에서 최소 대역폭은 반드시 만족되어야 하며 불필요한 대역폭의 사용을 방지하기 위한 최대 대역폭 이상의 입출력은 제한되어야 한다. 따라서, 최소 대역폭이 만족되지 않은 *L_MINBW_IO* 모드 상태를 갖는 제어 대상의 입출력 긴급도가 가장 크며, 최대 대역폭을 만족한 *L_NO_IO*의 모드 상태를 갖는 제어 대상의 긴급도는 고려할 필요가 없다. 최소와 최대 대역폭의 사이 값인 *L_RANGE_IO* 모드 상태를 갖는 제어 대상은 긴급도가 아주 낮으며 *L_MINBW_IO* 모드 상태를 갖는 제어 대상을 위하여 입출력을 양보하거나 잔여 대역폭을 사용하는 것이 바람직하다.

상기와 같이, 지역 입출력 모드는 제어 대상의 입출력 긴급도를 고려하기 위하여 최소 및 최대 대역폭을 경계로 하는 입출력 모드 상태들로 정의한 제어 대상의 속성이며, 지역 입출력 모드를 기반으로 차별적인 타임 슬라이스의 할당 정책을 운영한다

다음으로, 제어 대상이 갖는 지역 입출력 모드의 상태 변화에 따른 타임 슬라이스의 할당 방식에 대하여 살펴본다. 본 논문에서는 *L_DEFAULT_IO* 모드 상태에서 할당되는 타임 슬라이스를 기본 타임 슬라이스라고 하며 T_b로 정의한다. $M_{.total}$과 $m_{.total}$은 각각 전체 제어 대상의 최대 대역폭의 합과 최소 대역폭의 합을 의미하며 $Mm_{.total}$은 전체 제어 대상의 최대 대역폭과 최소 대역폭의 평균값이다. T_A는 제어 대상 A에게 할당되는 타임 슬라이스이며, M_A, m_A는 제어 대상 A에 설정된 최대 대역폭 및 최소 대역폭을 의미한다. 각 지역 입출력 모드에서의 타임 슬라이스의 할당 정책은 알고리즘 1과 같으며, N은 최적화 가능한 양수 값으로 정의된다.

제안하는 구간 제어 기법에서는 입출력의 제한을 위한 방식으로 그림 3에서와 같이 Sleep & Wakeup 방식을 사용하며, Sleep 상태로 전이된 제어 대상은 새로운 전역 타임 슬라이스가 시작되는 시점에 Wakeup 상태로 전이하며 입출력 수행을 재기하게 된다. 이 때, 정확한 Wakeup 시점을 계산하는 것은 중요하며 아래와 같은 방식으로 산정한다.

$$ST_A = T_{.ge} - T_C$$

ST_A는 제어 대상 A가 Sleep 상태에 있어야 하는 시간 간격을 의미하며 $T_{.ge}$는 전역 타임 슬라이스가 종료하는 시간이고 T_C는 현재의 시간, 즉 제어 대상 A가 Sleep 상태로 전이되는 시점을 의미한다.

(나) 전역 입출력 모드(Global I/O Mode)

전역 입출력 모드는 제어 디바이스 상에 존재하는

알고리즘 1. 구간 제어 기법에서의 타임 슬라이스 할당

setTimeSlice

Input : control group
Output : end point of time slice for control group
T_b : default time slice
$M_{.cg}$, $m_{.cg}$: maximum and minimum bandwidth value of control group, cg
$Mm_{.total}$: summary of the average values calculated by minimum and maximum bandwidth of all control groups

```
01: Integer ts, ts_end, current_time
02: get current time and save to current_time
03: if the local I/O mode of cg is L_MINBW_IO then
04:    ts = Tb + ( Tb * (( M.cg + m.cg ) / 2 ) / Mm.total )
05: else If the local I/O mode of cg is L_RANGE_IO then
06:    ts = Tb / N
07: else if the local I/O mode of cg is L_DEFAULT_IO then
08:    ts = Tb
09: else if the local I/O mode of cg is L_NO_IO then
10:    ts = 0
11: end if
12: ts_end = current_time + ts
13: return ts_end
```

제어 대상들이 갖는 지역 입출력 모드의 상태에 따라서 변화하는 속성이며, 제어 대상의 선정 방식을 결정하기 위한 수단으로 사용된다. 전역 입출력 모드는 *G_MINBW_IO*와 *G_LEFTOVER_IO*의 두 가지 모드 상태를 가지며 각각은 아래와 같이 정의된다.

- *G_MINBW_IO* : 제어 디바이스 상에 존재하는 제어 대상들 중에 *L_MINBW_IO*의 상태를 갖는 제어 대상이 하나 이상 존재하는 입출력 모드 상태
- *G_LEFTOVER_IO* : 제어 디바이스 상에 *L_MINBW_IO*의 상태를 갖는 제어 대상이 존재하지 않는 입출력 모드 상태

전역 입출력 모드에서 *G_MINBW_IO* 모드 상태는 제어 대상들 중에 최소 대역폭이 만족되지 않은 제어 대상이 존재함을 의미하므로, 다음 입출력 권한을 위한 제어 대상의 선정 시에 *L_MINBW_IO* 모드 상태를 갖는 제어 대상이 우선적으로 선정되어야 한다. 또한, *G_MINBW_IO* 모드 상태에 *L_MINBW_IO* 모드 상태를 갖는 제어 대상이 하나 이상 존재할 수 있으며 해당 제어 대상들 간의 우선순위 산정이 요구된다. *L_MINBW_IO* 모드 상태를 갖는 제어대상들 사이에서는 최소 대역폭을 만족하기 위한 잔여 대역폭이 많을수록 높은 우선순위를 갖는다. 아래의 식은 *G_MINBW_IO* 모드 상태에서의 제어 대상 선정을 위한 입출력 우선순위를 나타낸다.

$$L\_MINBW\_IO > L\_DEFAULT\_IO > L\_RANGE\_IO > L\_NO\_IO$$

G_LEFTOVER_IO 상태에서는 모든 제어 대상에 대하여 임의 방식(Random Scheduling)을 적용하여 선정하며, 선정된 제어 대상은 자신의 지역 입출력 모드에 따라서 타임 슬라이스를 할당 받는다.

그림 4는 전역 입출력 모드를 설명한 그림이며, G_A, G_B, G_C, G_D는 제어 디바이스의 대역폭을 공유하는 네 개의 서로 다른 제어 대상을 의미하고 G_B, G_C가 *L_MINBW_IO* 모드 상태에 존재하며 G_A가 *L_RANGE_IO* 모드 상태이고 G_D는 *L_DEFAULT_IO* 모드 상태이다. 따라서, 제어 대상의 선정에서 우선순위가 가장 높은 *L_MINBW_IO* 모드 상태의 G_B, G_C가 선택되고 최소 대역폭을 만족하기 위한 잔여 입출력 대역폭이 많이 남아있는 G_B가 가장 높은 우선순위를 가지게 되어서 다음 입출력을 위한 권한은 G_B에게 부여된다. 따라서, 네 개의 제어 대상 G_A, G_B, G_C, G_D의 우선순위는 아래와 같다.

$$G_B > G_C > G_D > G_A$$

3.2.2 상대적 대역폭을 제공하는 비율적 제어 기법 (Proportion Share Technique)

비율적 제어 기법은 제어 디바이스에서 제공하는 전체 대역폭의 크기에 상관없이, 가용한 대역폭을 제어 대상들 사이에 비율적으로 할당하기 위한 입출력 제어 방식이다. 앞 절에서 설명한 구간 제어 기법이 절대적, 정량적 입출력 제어 방식인 반면, 비율적 제어 기법은 제어 대상들의 서비스 중요도나 입출력 대역폭의 요구 사항에 기반한 상대적, 비율적 입출력 제어 방식으로 볼 수 있다. 비율적 제어 기법에서는 입출력 토큰이 입출력 권한을 의미하므로 많은 대역폭 비율을 할당 받은 제어 대상일수록 많은 토큰을 할당 받는다. 또한, 한 번에 할당 받는 입출력 토큰의 양은 제어 대상의 할당 대역폭 비율에 따라 다르지만, 할당 받은 입출력 토큰을 소진하는 횟수 또는 비

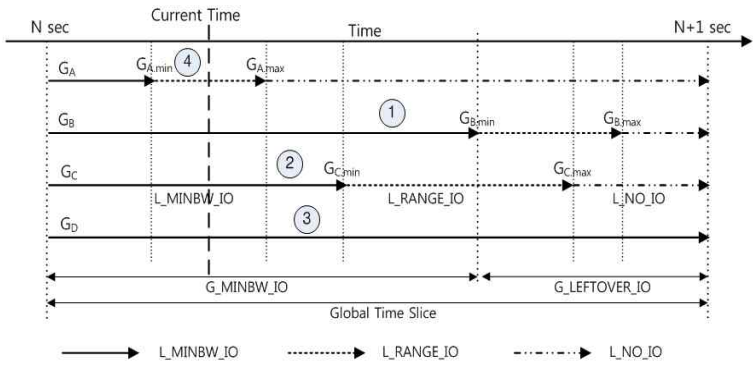

그림 4. 전역 입출력 모드 상태에 따른 제어 대상의 선정

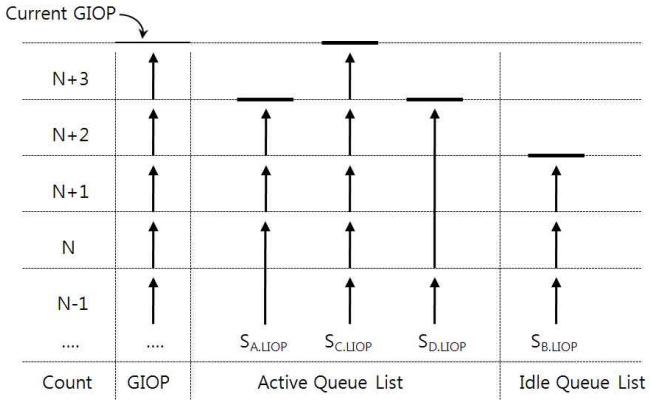

그림 5. 전역 및 지역 입출력 주기의 운영

율은 동일하게 유지되어야 한다는 제약 사항을 갖는다. 따라서, 제안하는 비율적 제어 기법에서는 상기 제약 사항을 만족시키기 위한 새로운 제어 수단으로 입출력 주기라는 것을 사용하며, 전역 입출력 주기와 지역 입출력 주기로 구분한다.

(가) 입출력 주기(I/O Period)

본 절에서는 전역 및 지역 입출력 주기의 운영 방법에 대하여 살펴보자. 전역 및 지역 입출력 주기의 갱신 여부 확인은 임의의 제어 대상에서 처리하여야 하는 입출력 요청의 수보다 잔여 토큰의 수가 적은 경우에 수행하며 전역 입출력 주기의 갱신 여부를 먼저 확인하고 지역 입출력 주기의 갱신 여부를 확인한다. 이 때, 전역 입출력 주기 값의 갱신은 제어 디바이스 상에 존재하는 모든 제어 대상을 상대로 입출력 토큰의 재충전을 허용한 횟수이며, 지역 입출력 주기 값의 갱신은 해당 제어 대상에게 실제로 토큰의 재충전을 수행한 횟수이다. 따라서, 개별 제어 대상의 입출력 토큰에 대한 재충전은 전체 제어 대상에 대한 입출력 토큰의 재충전을 허용한 횟수 내에서만 가능하다. 결과적으로, 전역 입출력 주기는 항상 지역 입출력 주기 보다 같거나 크며, 전역 입출력 주기보다 큰 지역 입출력 주기는 존재하지 않는다.

그림 5는 전역 및 지역 입출력 주기의 운영 방식을 나타내는 그림이며 *GIOP*는 전역 입출력 주기이고 *LIOP*는 지역 입출력 주기를 의미한다. 또한 *Count*는 전역 및 지역 입출력 주기가 갖는 주기 값이 된다. 네 개의 제어 대상 S_A, S_B, S_C, S_D가 존재하며 S_A, S_C, S_D는 Active Queue List[1]에 존재하고 S_B는 Idle Queue List에 존재한다.

그림 5와 같이, 전역 입출력 주기는 전역적으로 입출력 토큰의 재충전을 허용한 횟수이므로 항상 1씩 증가하며, 지역 입출력 주기는 각 제어 대상의 입출력 수행 상태에 따라 가변적으로 증가한다. 즉, 지역 입출력 주기는 갱신 시에 전역 입출력 주기가 갖는 주기 값으로 설정되며 전역 입출력 주기 값과 지역 입출력 주기 값의 차이는 해당 제어 대상이 전역적인 토큰의 재충전에 참여하지 못한 횟수를 의미한다. 예를 들어, 제어 대상 S_D의 지역 입출력 주기는 N에서 $N+3$으로 한 번에 3만큼 갱신이 이루어졌으며 이는 S_D가 입출력 요청의 부재로 Idle Queue List에 있는 동안 세 번의 전역적인 입출력 토큰의 재충전에 참여하지 못하였음을 의미한다. 하지만, 입출력 요청이 발생함에 따라서 Active Queue List로 이동하면서 갱신 시점의 전역 입출력 주기 값인 $N+3$으로 갱신된 경우이다. 1씩 증가하는 지역 입출력 주기를 갖는 제어 대상 S_C는 지속적인 입출력 요청이 존재하는 경우이며, 전역 입출력 주기를 갱신시키는 제어 대상이 된다.

상기와 같이, 재충전이 허용된 입출력 토큰을 사용하지 못한다는 것은 할당된 대역폭의 비율에 오차가 발생함을 의미하며, 제안하는 비율적 제어 기법에서는 할당 대역폭의 비율에 발생하는 오차를 보정하기 위한 방법으로 입출력 토큰의 이월이라는 개념을 적용한다. 이는 제어 대상의 입출력 토큰에 대한 재충전

1) Active Queue List에는 입출력 요청이 존재하는 제어 대상의 큐가 관리되고 Idle Queue List에는 입출력 요청이 존재하지 않는 제어 대상의 큐가 관리되는 구조이며, 입출력 유무에 따라서 제어대상의 큐는 동적으로 이동된다.

시에, 전역적인 입출력 토큰의 재충전에 참여하지 못한 횟수만큼 입출력 토큰을 보상함으로써 할당 대역폭의 비율에 대한 오차를 보정하기 위함이다. 하지만, 입출력 요청이 지속적으로 존재하지 않는 제어 대상은 입출력 토큰의 반복적인 이월이 발생하는 문제점을 갖는다. 따라서, 이러한 문제점을 보정하기 위한 수단으로 이월할 수 있는 주기의 수를 제한하는 이월 상한 값을 사용하며 적용하는 응용 및 서비스의 입출력 형태에 따라 적절한 값의 설정이 가능하다. 본 논문의 성능평가에서는 이월 상한 값으로 2를 사용하며 두 번까지 입출력 토큰의 이월을 허용하고 있다.

(나) 입출력 토큰(I/O Token)의 관리

특정 제어 대상이 한 번의 입출력 주기에서 사용할 수 있는 입출력 토큰의 수는 제어 대상에게 할당된 대역폭의 비율과 비례한다. 즉, 매 입출력 주기에서 할당할 수 있는 전체 입출력 토큰의 수가 지정되어 있으며, 제어 대상에게 할당된 대역폭의 비율에 따라 아래와 같이 입출력 토큰이 분배된다.

*cg.weight*는 특정 제어 대상 *cg*에 대한 대역폭 할당 비율이며, *total_weight*는 모든 제어대상에게 할당된 대역폭 비율의 합이다.

$$TOTAL\_TOKEN * (cg.weight / total\_weight)$$

알고리즘 2는 제어 대상에 대한 입출력 토큰의 재충전 방식을 설명하는 알고리즘이며, 제어 대상에 따른 기본 입출력 토큰(*cg.token_initial*)을 기반으로 입출력 토큰의 재충전 양을 산정하고 있다. *diff*는 이월 상한값이 적용된 전역 입출력 주기와 지역 입출력 주기의 차이이며 라인 08은 제어 대상 *cg*가 참여하지 못한 전역적인 입출력 토큰의 재충전 횟수만큼 기본 입출력 토큰을 이월하는 부분이다. 마지막으로, 라인 10과 11에서 지역 입출력 주기의 갱신에 따른 기본 입출력 토큰 수에 이월되는 토큰의 수를 합하여 재충전되는 총 토큰의 수인 *cg.token*을 반환하고 있다.

제안하는 비율적 제어 기법에서의 입출력 우선순위는 가용한 토큰이 많은 제어 대상일수록 높은 우선순위를 산정하는 정책을 사용한다. 가용 토큰이 많다는 것은 오랫동안 입출력 수행에 참여하지 못한 제어 대상이거나 서비스 중요도나 입출력 요구사항이 높아서 많은 입출력 대역폭을 할당 받은 제어 대상이기 때문이다. 따라서, 제어 대상의 선정을 위한 입출력 우선순위는 해당 제어 대상이 현재 사용할 수 있는 가용한 토큰의 총수를 계산하여 산정되며, 이는 입출력 토큰의 재충전 시에 수행하였던 재충전 토큰 수의 산정 방식과 동일하다. 또한, 산정된 총 가용 토큰 수를 각 제어 대상의 기본 토큰 수로 나누어 줌으로써 각 제어 대상 입장에서의 입출력 긴급도를 고려한다.

4. 성능 평가

본 장에서는 제안하는 입출력 제어 방식에 대한 성능 평가를 수행하며, 성능을 검증하기 위하여 제안 입출력 방법의 수행 부하 평가, 입출력 대역폭의 균등 분할 성능 평가 및 차등 분할 성능 평가로 구분한다.

그림 6은 제안하는 입출력 제어 방식의 성능 평가를 위한 기본적인 시스템 환경을 도시한 그림이며, 성능 평가를 위하여 사용자 영역의 서비스 개수를 변화시키거나 서비스가 포함하는 입출력 프로세스

알고리즘 2. 비율적 제어 기법에서 입출력 토큰의 재충전

```
refillToken( cg )
Input :   control group
Output :   the number of tokens to be refilled
GIOP : Global I/O Period, LIOP.cg : Local I/O Period of control group, cg
01:  Integer refill_token, diff
04:  diff = GIOP - LIOP.cg
05:  if diff is zero then
06:     refill_token = NULL
07:  elseif
08:     refill_token = cg.token_initial * diff
09:     decrease the number of left global token by refill_token
11:  endif
10:  cg.token += refill_token
12:  return cg.token
```

표 3. 성능 평가 환경

항 목	특 징	비 고
시스템	Samsung Smart Server, ZSS108	
운영체제	CentOS-5.3, kernel-2.6.32	
프로세서	Intel Pentium(R) 4, 3.20GHz	2 Core
메모리	2Gbytes	1G * 2
디스크	Samsung HD322HJ, 7200RPM, 320Gbytes	S-ATA 2 방식
시험 도구	fio-1.26	Flexible I/O Tester
구간 제어 기법의 설정값	G_MINBW_IO : HZ/10 G_RANGE_IO : HZ/50	기본 타임 슬라이스(Tb)
비율적 제어 기법의 설정값	TOTAL_TOKEN : 2048	전역 주기에서의 전체 할당 토큰

표 4. 입출력 제어 기법에 따른 총 입출력 대역폭의 비율

	일반환경	구간 제어 기법	비율적 제어 기법
총 대역폭 비율	1	0.947	0.952

의 개수를 변화시킴으로써 평가를 수행한다. 사용되는 입출력 프로세스는 fio[2]라는 입출력 성능 측정 도구로서 하나의 fio 수행으로도 전체 디스크 대역폭을 점유할 수 있는 도구이다. 본 장의 차등 분할 성능 평가에서는 100개씩의 fio를 포함하는 3개의 서비스를 생성하여 성능 평가를 수행한다. 제안하는 입출력 제어 방식의 성능 평가를 위한 환경은 표 3과 같으며, 결과 값에 영향을 줄 수 있는 하드웨어 측면의 환경과 소프트웨어 측면의 환경을 고려하며 각 입출력 제어 기법에서의 중요 설정값을 포함한다.

표 3에서 구간 제어 기법의 기본 타임 슬라이스

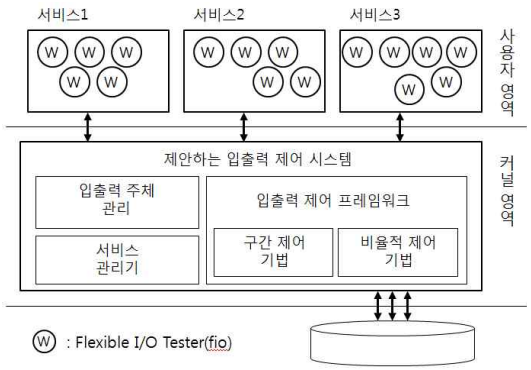

그림 6. 입출력 제어 기법의 성능 평가 환경

2) fio는 디스크 입출력 성능 평가를 위한 공개SW 도구로서 http://freshmeat.net/projects/fio/ 에서 참조가능함

(*Tb*)값과 비율적 제어 기법의 전체 할당 토큰(*TOTAL_TOKEN*) 값은 입출력의 제어 시에 제어 대상에 대한 스케쥴링 빈도를 결정하므로 전체 대역폭의 크기에 영향을 미치는 요인으로 작용한다.

4.1 입출력 제어의 수행 부하 평가

본 절에서는 제안하는 입출력 제어 방식의 수행 부하를 살펴본다. 평가는 제안하는 입출력 제어 방식을 적용하지 않은 상태에서의 전체 대역폭인 “일반환경”의 대역폭을 측정하고 구간 제어 기법을 적용한 경우의 대역폭인 “구간 제어 기법”의 대역폭과 비율적 제어 기법을 적용한 경우의 대역폭인 “비율적 제어 기법”의 대역폭을 각각 측정하여 비교한다.

그림 7의 그래프는 제어 대상 수의 증가에 따른 입출력 제어 기법의 수행 부하를 측정한 결과이며, 1개의 제어 대상부터 10개의 제어 대상에 대한 입출력 제어 시의 수행 부하를 측정 하였다.

각 그래프는 해당 시험에서 제어 대상들이 사용한 전체 입출력 대역폭의 합을 의미하며 일반 환경에서의 전체 대역폭과 각 입출력 제어 기법에서의 전체 입출력 대역폭의 차가 수행 부하를 나타낸다. 본 시험에서는 입출력 캐시나 디스크 자체 캐시의 영향을 최소화하기 위하여 “*Direct I/O-Sequential*” 형태의 입출력을 사용하였으며, 각 제어 대상은 서로 다른 파일에 대한 입출력을 수행한다. 표 4는 기준 대역폭인 일반 환경에서의 전체 대역폭을 1로 변환하였을 때,

각 입출력 제어 기법에 의한 전체 대역폭 비율을 나타낸 표이며 제안 입출력 제어 방법에 의한 입출력 제어의 수행 부하는 약 5% 정도임을 확인할 수 있다.

4.2 입출력 대역폭의 균등 분할 성능 평가

본 절에서는 입출력 대역폭의 균등 분할에 대한 성능 평가를 수행한다. 입출력 대역폭의 균등 분할에 대한 평가에서는 균등 분할 기능을 제공하는 리눅스 운영체제의 입출력 제어기인 CFQ(Complete Fare Queuing) 제어기와 제안하는 입출력 제어 방법의 성능을 비교 평가한다. 그림 8은 세 개의 프로세스에 대한 입출력 균등 분할 성능 평가의 결과이며, 좌측이 CFQ에 의한 결과이고, 오른쪽이 제안 방식에 의한 입출력 대역폭의 균등 분할 결과이다. 그림에서 나타내는 바와 같이, CFQ에 의한 입출력 대역폭의 균등 분할 결과는 다소 큰 대역폭의 변동이 발생하는 반면, 제안 방식에 의한 입출력 대역폭의 균등 분할에서는 세 개의 프로세스들이 비교적 안정적으로 대역폭을 공유하고 있음을 확인할 수 있다. 따라서, 제안하는 방법은 대등한 입출력 중요도를 갖는 여러 제어 대상들 사이에 안정적인 입출력의 균등분할을 지원함으로써 성능 안정성을 보장함을 확인할 수 있다.

4.3 입출력 대역폭의 차등 분할 성능 평가

본 절에서는 입출력 대역폭의 차등 분할에 대한 제어 성능을 검증하기 위한 평가를 수행하고 결과를 분석한다. 본 평가는 여러 서비스에 차등적인 대역폭을 할당하고 각 서비스에 다수의 입출력 프로세스들을 추가하였을 때, 각 서비스에 할당된 입출력 대역폭이 상호간에 격리됨을 검증하기 위한 성능 평가로

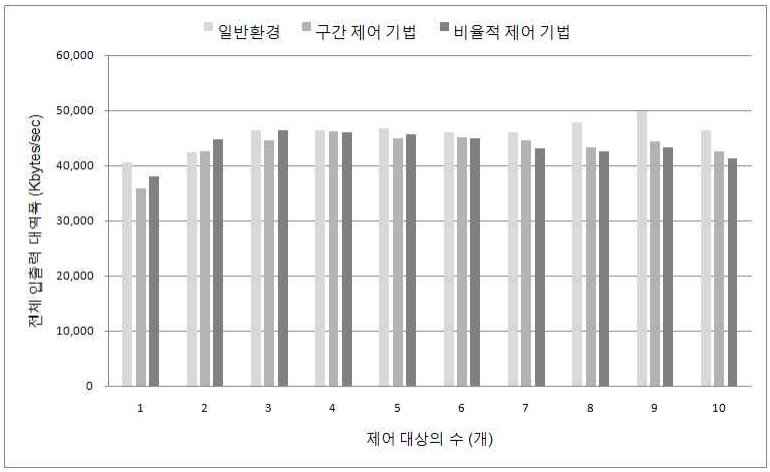

그림 7. 제안하는 입출력 제어 방식의 수행 부하 평가 결과

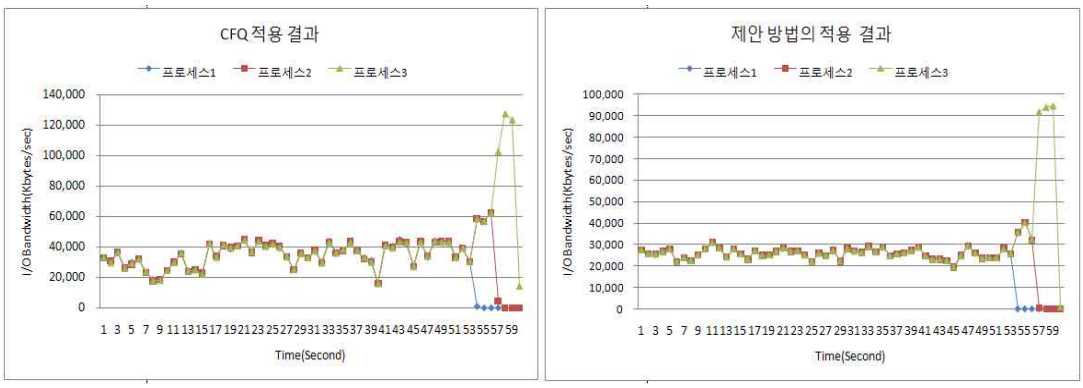

그림 8. 입출력 대역폭의 균등 분할 성능 평가 결과

설정값은 각각 표 5, 표 6과 같다.

본 성능 평가에서는 세 개의 제어 대상인 서비스 1, 서비스 2, 서비스 3에 각각 100개씩의 입출력 프로세스들이 구동할 때, 서비스 상호간에 할당된 입출력 대역폭의 격리 여부를 검증한다. 따라서, 총 300개의 입출력 프로세스들이 동시에 구동되며, 구간 제어 기법 및 비율적 제어 기법에 의한 시험 결과값은 그림 9, 그림 10에서 보여주는 바와 같다.

시험의 결과는 전반적으로 각 입출력 제어에 대한 대역폭의 변동이 심하게 나타나고 있음을 확인할 수 있다.

비율적 제어 기법의 전체 입출력 대역폭이 구간 제어 기법과 비교하여 적은 것은 시험 설정값에서 기술한 바와 같이, 전체 입출력 토큰을 2,048개로 고정하였기 때문이다. 하지만, 대역폭의 공유 측면에서는 할당된 비율에 근접하여 지원되고 있음을 확인할 수 있다. 반면, 구간 제어 기법의 경우는 정량적인 대역폭을 제공하기 위한 것으로 최소 대역폭이 만족되지 않으면 심각한 문제를 유발할 수 있으므로 지정된 입출력 대역폭이 반드시 제공되어야 하며, 성능 평가 결과에서 나타낸 바와 같이, 입출력 제어 결과의 대부분은 최소 대역폭을 만족하면서 서비스들에게 예측 가능한 정량적 대역폭을 제공하고 있음을 확인할 수 있다.

표 5. 구간 제어 기법을 위한 입출력 대역폭의 차등 분할 평가 설정값

제어 기법	연산 종류	버퍼 사용 여부	입출력 패턴	서비스 ID	할당 대역폭	I/O 프로세스 개수
구간 제어 기법	Write	Delayed I/O	Sequential I/O	1	12500 : 13500	100
				2	8000 : 9000	100
				3	3500 : 4500	100
			Random I/O	1	4500 : 5500	100
				2	2667 : 3667	100
				3	833 : 1833	100
		Direct I/O	Sequential I/O	1	10500 : 11500	100
				2	6667 : 7667	100
				3	2833 : 3833	100
			Random I/O	1	300 : 405	100
				2	170 : 270	100
				3	35 : 135	100

표 6. 비율적 제어 기법을 위한 입출력 대역폭의 차등 분할 평가 설정값

제어 기법	연산 종류	버퍼 사용 여부	입출력 패턴	서비스 ID	할당 대역폭	I/O 프로세스 개수
비율적 제어 기법	Write	Delayed I/O	Sequential I/O	1	50%	100
				2	33.33%	100
				3	16.66%	100
			Random I/O	1	50%	100
				2	33.33%	100
				3	16.66%	100
		Direct I/O	Sequential I/O	1	50%	100
				2	33.33%	100
				3	16.66%	100
			Random I/O	1	50%	100
				2	33.33%	100
				3	16.66%	100

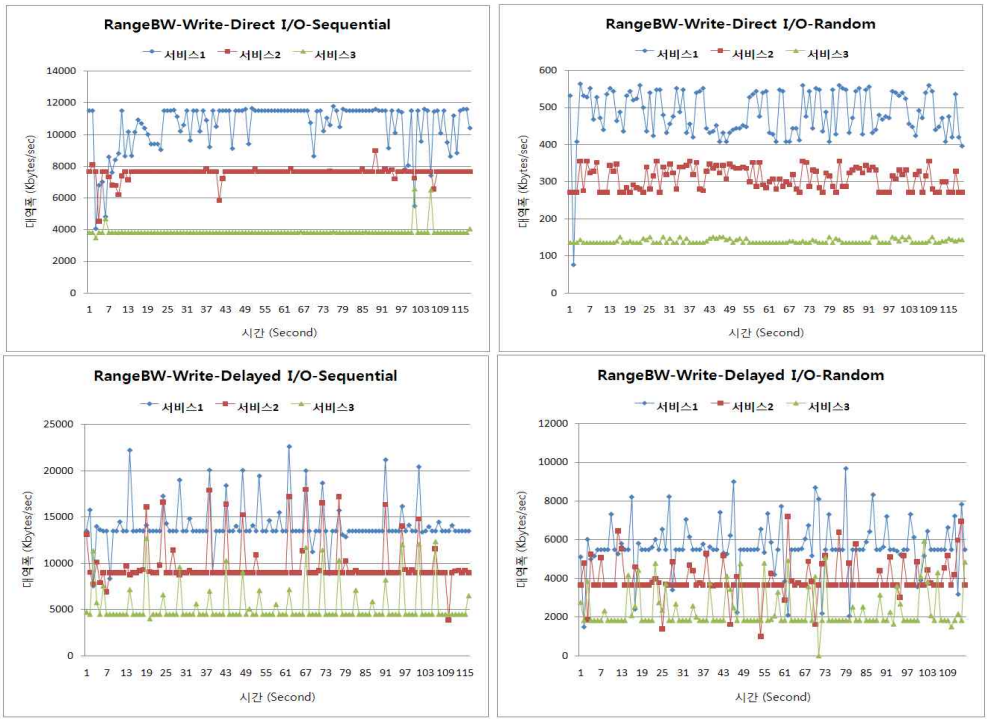

그림 9. 구간 제어 기법에 의한 입출력 대역폭의 차등 분할 성능 평가 결과

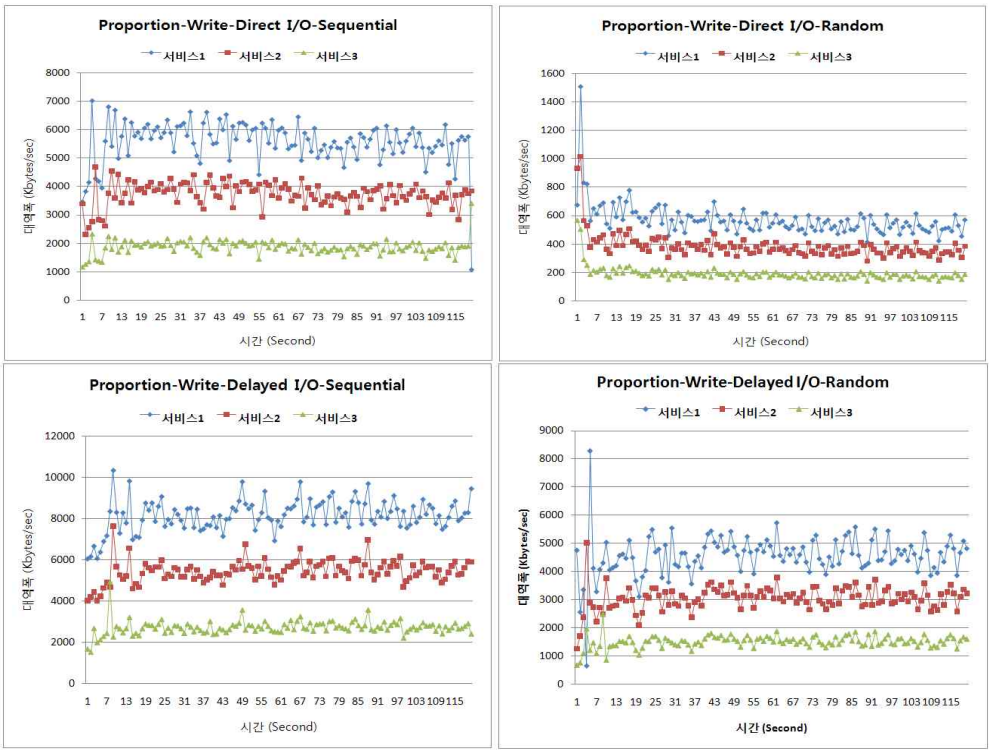

그림 10. 비율적 제어 기법에 의한 입출력 대역폭의 차등 분할 성능 평가 결과

5. 결 론

본 논문에서는 예측 가능한 대역폭을 제공하는 서비스 기반 디스크 입출력 제어 방식을 제안하였다. 제안한 입출력 제어 방식은 제한된 입출력 자원 환경에서 특정 서비스의 품질 및 성능 안정성을 보장하기 위한 것으로, 본 논문에서는 상기 목적을 달성하기 위하여 프로세스 그룹을 하나의 관리 객체로 제공하는 서비스 관리기, 예측 가능한 대역폭의 제공을 위한 구간 제어 기법과 비율적 제어 기법 및 제어 정확도 향상을 위하여 지연 입출력 요청의 주체 관리 기법을 제안하였다.

제안하는 입출력 제어 방식은 성능 평가 결과에서 제시한 바와 같이, 입출력 대역폭의 균등 분할을 위한 시험 결과에서는 기존 운영체제의 입출력 제어기인 CFQ보다 우수한 성능을 보여주었다. 또한, 입출력 대역폭의 차등 분할 시험에서는 수 백 개의 프로세스가 구동하는 환경에서 구간 제어 기법 및 비율적 제어 기법을 통한 예측 가능한 대역폭을 제공하였다.

따라서, 제안한 입출력 제어 방식은 서비스 기반의 예측 가능한 입출력 대역폭을 제공함으로써, 안정적인 서비스 품질 및 성능을 유지하는데 효과적이며, 제한된 입출력 자원의 효율적인 사용을 가능하게 하였다.

본 논문의 향후 연구로는, 최근 이슈가 되고 있는 SSD(Solid State Disk)와 같은 메모리 기반 디스크에서 제안하는 입출력 제어 방법을 적용하기 위한 연구가 필요하며, 서비스의 성능 안정성을 효과적으로 지원하기 위하여 CPU, Memory 및 네트워크 자원 관리와의 연동 및 일관된 자원 관리를 지원하는 통합 자원 관리 시스템의 연구가 요구된다.

참 고 문 헌

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing,” *Technical Report*, EECS Department, University of California, Berkeley, 2009.

[2] M.A. Rappa, “The utility business model and the future of computing services,” *IBM Systems Journal*, Vol. 43, No. 1, 2004.

[3] L. Eggert and J. Heidemann, “Application-level differentiated service for Web servers,” *Journal of the World Wide Web(Springer Netherland)*, pp. 133-142, 2006.

[4] J. Axboe, “Linux Block IO-present and future,” Proceedings of the Linux Symposium, pp. 51-61, 2004.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R.. Neugebauer, I. Pratt and A. Warfield, “Xen and the art of virtualization,” Proceedings of the 19th ACM symposiumon Operating Systems Principles, pp. 164-177, 2003.

[6] J. Nieh and O. C. Leonard, “Examining VMware,” *Dr. Dobb's Journal*, 2000.

[7] Paul B. Menage., “Adding Generic Process Containers to the Linux Kernel,” Proceedings of Ottawa Linux Symposium, pp. 45-57, 2007.

[8] S. Nagar, H. Franke, J. Choi, M. Kravetz, C. Seetharaman, V. Kashyap and N. Singhvi, “Class-based Prioritized Resource Control in Linux,” Proceedings of the Ottawa Linux Symposium, pp. 150-168, 2003.

[9] S. Watters, “Linux Process Aggregates(PAGG),” SGI, http://oss.sgi.com/projects/pagg

[10] S.Uchida, “Yet Another I/O Bandwidth Controlling Subsystem for Cgroup based on CFQ,” http://lwn/Articles/275994,2008.

[11] V. Tarasov, “I/O bandwidth controlling subsystem for CGroups based on CFQ,” http://thread.gmane.org/gmane.linux.kernel/656570/focus=656573

[12] A. Righi, “cgroup: block device I/O controller(v8),” http://thread.gmane.org/gmane.linux.kernel.containers/5975,2008.

강 동 재

1999년 인하대학교 전자계산공학과 (공학사)
2001년 인하대학교 전자계산공학과 (공학석사)
2010년 인하대학교 정보공학과 (공학박사)
2001년~현재 한국전자통신연구원 선임연구원
관심분야 : 데이터베이스, 자료저장시스템, 공개SW 기반기술, 운영체제, 시스템SW, 클라우드컴퓨팅, 고성능 컴퓨팅

이 평 화

2009년 세종대학교 소프트웨어공학과 (공학사)
2009년~2010년 한국전자통신연구원 인턴연수생
2010년~현재 한양대학교 전자컴퓨터통신공학과 석사과정
관심분야 : 운영체제, 임베디드 시스템, 시스템SW, 클라우드컴퓨팅

정 성 인

1987년 부산대학교 전자계산학과(이학사)
1989년 부산대학교 전자계산학과(이학석사)
2006년 충남대학교 컴퓨터공학과(공학박사)
1999년~2000년 미국 SCO 방문연구원
2004년~현재 한.중.일 공개SW포럼 WG1 한국의장
1989년~현재 한국전자통신연구원 책임연구원
관심분야 : 운영체제커널, 공개S/W, 시스템관리, 고가용성, 컴퓨터구조, 고성능 컴퓨팅, 클라우드컴퓨팅