대리인에 의한 캐쉬 일관성 유지 방식에서의 클라이언트들간의 상호 협조

김 경 자, 장 태 무
동국대학교 컴퓨터공학과

A Cooperation with Clients on Agent-Initiated Cache Consistency Sheme

KyoungJa Kim, TaeMu Chang
Dept. of Computer Engineering, Dongguk University

요 약

분산 파일 시스템에서는 파일 서버는 사용률이 높고, 통신 네트워크나 클라이언트 워크스테이션은 상대적으로 낮은 것으로 보고 하고 있다. 따라서, 시스템의 확장은 파일 서버가 처리해야 할 작업 부담을 증가하는 결과를 초래하므로 파일 서버는 시스템 상의 주된 병목 구간이 된다. 따라서 분산 시스템의 성능을 향상시키기 위해서는 파일 서버의 작업 부담을 클라이언트로 이전시키고, 가급적이면 파일 서버가 유지하는 상태 정보의 양을 줄여야 한다.

본 논문에서는 파일 서버에서 유지되는 상태 정보의 양이 파일 서버의 작업 부담에 비례한다는 가정 하에 파일 서버의 작업 부담을 클라이언트들에게 이양시키는 대리인에 의한 캐쉬 일관성 유지 방식과 클라이언트들간의 상호 협조를 통해 요구된 파일에 대한 응답 시간을 줄이는 방안을 제안하였다.

1. 서 론

기존의 연구에 의하면, 분산 파일 시스템은 서버의 사용률이 높고, 통신 네트워크나 클라이언트 워크스테이션은 상대적으로 사용률이 낮은 것으로 나타난다[1,2]. 따라서 시스템의 확장은 파일 서버가 처리해야 할 작업 부담을 증가하는 결과를 초래하므로 파일 서버는 시스템 상의 주된 병목 구간이 된다. 이러한 이유에서 볼 때, 분산 시스템의 확장성을 향상시키기 위하여 서버의 부담을 클라이언트로 이전시켜 파일 서버가 유지하는 상태 정보를 줄여야 한다[3,4]. 이와 같은 방법으로 전체 사용률을 줄이는 방식이 대리인에 의한 일관성 유지 방식에서 제안되었다[7]

본 논문에서는 더 나아가 클라이언트들간의 협동을 함으로 서버나 대리인과의 빈번한 접근 횟수를 줄이고, 클라이언트에서 요구된 파일의 응답 시간을 줄이고자 한다. 빈번한 파일 서버와의 교신을 줄이기 위해 대리인와 하위의 클라이언트들은 전체 파일 캐싱 방법을 씀으로 해서, 블록 캐싱 방법보다는 파일 서버의 재접근 가능성을 최소화한다.

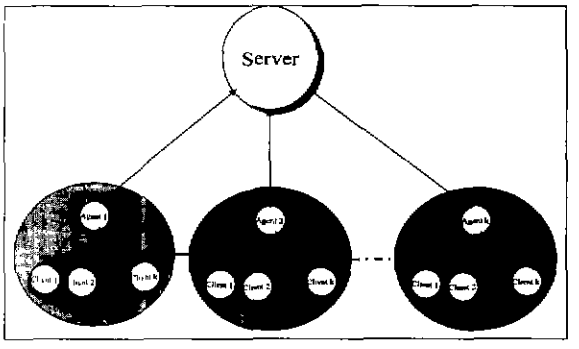
본 논문은 2장에서 대리인 방식에서의 전체적인 계층 구조와 각 계층의 역할에 대해서 개략적으로 살펴보고, 3장에서는 각 클라이언트들이 파일을 개설하는데 있어서 어떠한 과정을 거치는지에 대해서 살펴본다 마지막으로 4장에서는 결론과 향후 연구방향에 대해서 기술한다.

2. 대리인에 의한 캐쉬 일관성 방식

2.1 대리인 방식에서의 전체적인 계층 구조

본 논문에서 대리인에 의한 캐쉬 일관성 유지 방식에서 제안한 구조가 적용된다. 파일 서버의 작업 부담을 줄이기 위하여 각각의 파일에 대해 [그림 1]과 같은 논리적인 공유 구조를 형성한다. 즉, 임의의 파일을 공유하고 있는 클라이언트의 개수가 N일 때, 그 중 한 클라이언트를 대리인으로 선정하여 파일 서버와 교신하게 하고, 나머지

(N-1)개의 클라이언트들은 파일에 대한 내용을 얻기 위해 파일 서버와 직접 교신하지 않고, 선정된 대리인에게 자료를 요청하는 방식을 채택하였다. 따라서 각각의 공유 파일에 대해 하나의 대리인이 존재하게 된다.



[그림 1] 대리인 방식의 전체적인 계층 구조

다음은 각 계층별로 파일 서버, 대리인과 클라이언트의 역할에 대해 간략하게 알아본다.

2.1.1. 파일 서버의 역할

서버는 각각의 공유 파일에 대해 단지 대리인에 대한 정보만을 유지하고 있다. 또한, 파일 서버의 역할은 대리인의 선정과 선정된 대리인으로부터 오는 자료의 요청을 처리해 주어야 하고, 공유 파일을 사용하고자 하는 새로운 클라이언트에게 대리인을 소개시켜 주는 역할을 수행한다.

2.1.2. 대리인의 역할

대리인은 공유 파일을 사용중인 클라이언트들에 대한 ID, 가장 최근에 파일을 갱신한 클라이언트 ID와 일관성을 유지하기 위한 최근의 버전 번호를 유지해야만 한다. 선정된 대리인은 해당 공유 파일에 대해서 다른 클라이언트로부터 오는 자료 요청을 처리해 주어야 하고, 또한 파일의 내용이 변경이 되면 수정된 버전 번호와 해당 파일을 수정한 클라이언트 ID를 하위의 공유 파일을 사용하고 있는 클라이언트들에게 Broadcasting함으로서 내용이 수정되었다는 메시지를 보냄으로 파일의 일관성이 유지될 수 있도록 하는 책임을 갖는다.

2.1.3. 클라이언트의 역할

각각의 클라이언트들은 공유 파일에 대해 가장 최근의 자료를 가지고 있는 클라이언트 ID와 버전번호를 대리인에게서 받는다. 자료 요청이 필요한 경우에 최근의 버전 번호를 가지고 있는 클라이언트에게 파일을 요청하여 서비스 받는다.

3. 클라이언트들간의 협동 과정

3.1 클라이언들간의 협동의 필요성

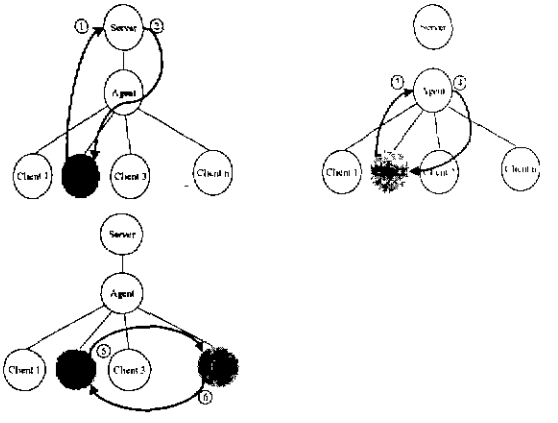
클라이언트들간의 상호 협조를 통해 작업 부담이 많은 파일 서버보다는 작업량이 적은 클라이언트에게 자료를 서비스 받음으로 서 자료의 응답 시간을 줄일 수 있다.

3.2 임의의 클라이언트에서의 파일 개설 과정

임의의 클라이언트에서 파일을 처음 개설한 경우와 재개설하는 경우, 또한 읽기 위한 개설과 쓰기 위한 개설의 경우에 따라 약간씩 과정의 차이가 있다. 경우에 따른 차이를 다음과 같이 세가지로 나누어 생각하였다.

3.2.1 파일을 처음 개설한 경우

[그림 2]는 임의의 클라이언트가 파일을 개설하는 과정을 나타낸다. 임의의 클라이언트 2가 공유 파일을 개설하고자

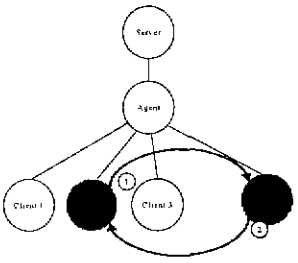


[그림 2] 임의의 클라이언트에서의 파일 개설

할 때, 클라이언트 2는 해당 파일에 대한 파일 서버에게 개설 요청을 한다(과정 ①) 이 요청을 받은 파일 서버는 해당 파일에 대한 대리인 ID를 제공한다(과정 ②) 대리인 ID를 받은 클라이언트 2는 해당 대리인에게 다시 개설 요청을 한다(과정 ③). 개설 요청을 받은 대리인은 자신의 캐쉬에 개설 요청을 한 클라이언트의 ID를 적재하고 가장 최근의 자료를 가지고 있는 클라이언트와 버전 번호를 넘긴다(과정 ④). 클라이언트 2는 대리인에게서 넘겨 받은 클라이언트에게 자료 요청을 한다(과정 ⑤). 요청을 받은 클라이언트는 클라이언트 2에서 요청한 자료를 서비스한다(과정 ⑥).

3.2.2 파일을 읽기 위해 재개설한 경우

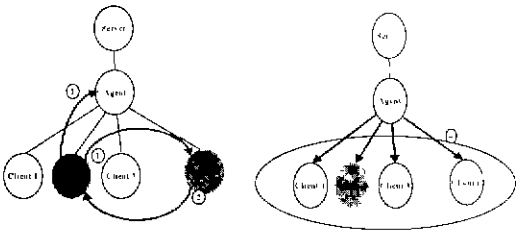
[그림 3]과 같이 이미 개설된 상태에서 재개설을 하고자 할 경우에는 이전의 대리인에게서 받은 클라이언트 ID를 가지고, 파일서버나 대리인에게 문의하는 과정을 없이 바로 마지막 클라이언트에게 파일을 요청한다. 그러므로, 재개설의 경우는 서버, 대리인의 접근이 없음으로 전체적인 네트웍 통신량의 횟수를 줄일 수 있다. 파일을 읽기 위해 재개설하는 경우는 파일을 처음 개설할 때, 대리인에게서 받은 마지막 내용을 가지고 있는 클라이언트에게 파일을 요청한다(과정 ①). 이 요청을 받은 클라이언트는 자료를 요청한 클라이언트에게 해당 파일을 서비스한다(과정 ②). 파일을 읽기 위해 재개설하는 경우는 [그림 3]과 같이 두 번의 과정만을 거치면 된다.



[그림 3] 파일을 읽기 위해 재개설한 경우

3.2.3 파일을 쓰기 위해 재개설한 경우

[그림 4]와 같이 클라이언트에서 파일을 읽기 위해서 개설하는 경우와는 다르게, 캐쉬 일관성을 유지 하기 위한 방법으로 버전 번호를 이용해, 쓰기 위해서 개설한 경우에는 버전 번호를 증가함으로 서 더욱 최근의 것임을 알린다. 임의의 클라이언트 2에서 해당 파일을 쓰기 위해서 개설한 경우에는 다음과 같은 순서로 해당 파일을 서비스 받을 수 있다. 임의의 클라이언트 2는 해당 파일을 마지막 클라이언트에게 요청을 한다(과정 ①).이 요청을 받은 클라이언트는 요청한 자료를 클라이언트 2에게 제공한다(과정 ②).자료를 제공받은 클라이언트 2는 내용을 수정한 후 대리인에게 파일의 내용이 변경 됨을 알리기 위해 새로운 버전 번호와 자신의 ID를 넘겨준다(과정 ③). 버전 번호와 클라이언트 ID를 넘겨 받은 대리인은 자신의 캐쉬에 새로운 버전 번호와 클라이언트 2의 ID를 적재하고, 하위의 다른 모든 클라이언트들에게 파일의 내용이 변경 됨을 알리기 위해 새로운 버전 번호와 최근 수정한 내용을 가지고 있는 클라이언트 2의 ID를 Broadcasting함으로 내용이 변경된 것을 알린다(과정 ④).



[그림 4] 파일을 쓰기 위해 재개설한 경우

4. 결론 및 향후 연구방향

분산 파일 시스템에서의 서버의 사용률이 높고, 통신네트워크나 클라이언트 워크스테이션은 상대적으로 사용률이 낮은 것으로 나타난다. 따라서 본 논문은 서버의 작업 부담을 클라이언트로 이전시키고, 클라이언트들간의 상호 협조를 통해 서버와 대리인의 상태 정보의 갱신 횟수를 줄이고, 또한 응답 시간을 줄이는 방안을 제시하였다. 향후에는 대리인 방식의 가장 큰 단점으로 파일 서버로부터 소개 받은 대리인이나 클라이언트가 네트워크 구성상 접근할 수 없을 때나, 대리인이나 클라이언트가 위치한 워크스테이션이 고장이 났을 경우에는 클라이언트가 공유 구조를 접근할 경로를 잃어버리게 된다는 것이다

참 고 문 헌

[1] Lazowska, E. D., Zohorjan, J, Cheriton, D. R and Zwaenepoal, W, "File Access Performance of Diskless Workstation" ACM TOCS Vol 4 No 3, pp 238-268, 1986

[2] Ousterhout, J K., da Costa, H Harisson, D., Kunze J A, Kupfler M, and Tompson J g, "A trace-driven Analysis of the 4 2 BSD Unix File system" Proc 10th ACM SOSP pp 15-24, 1985.

[3] Blaze, M. and Alonso, R.., "Dynamic Hirarchical Caching in Large-Scale distributed File System," Proc 12th ACM DCS. pp.521-528

[4] Satyanarayanan, M "The Influence of Scale on Distributed File System Design," IEEE TOSE, Vol. 18, No 1, pp.1-8, 1992

[5] Nelson. M., Welch B, and Ousterhout. J. K.., "Caching in the Sprite Network File System". ACM TOCS VOL 6 No. 1, pp.134-154. 1988.

[6] 홍봉기, 장태무, "분산 파일 시스템의 확장성을 위한 대리인-시작 캐쉬 일관성 유지 방식", 한국정보과학회, 정보과학논문지(A), 제23권, 제2호,pp.162-179. 1996.