DBpia

비연결 수행 이동컴퓨팅 태스크의 성능 분석

정회원 정 승 식*, 김 재 훈*

Performance Analysis of Disconnected Operation on Mobile Computing

Seong-Sik Jung*, Jai-Hoon Kim* *Regular Member*

요 약

무선 네트워을 포함하는 이동컴퓨팅에서는 무선 통신망의 특성상 잦은 끊김과 높은 에러율 때문에 비연결시에도 수행을 계속할 수 있는 기능이 필요하다. 이러한 이동컴퓨팅 환경에서 비연결 수행기능을 제공하기 위해 많은 개념과 이론들이 제안되었다. 본 논문에서 이동 컴퓨팅 환경에서 비연결 수행상태를 포함한 태스크의 평균 수행시간을 분석하였다. 비연결 수행 태스크는 데이터 호딩(Data Hoarding), 비연결 수행(Disconnected operation), 블록(Block)의 3가지 상태로 분리할 수 있다. 이러한 3가지 상태에서 여러 가지 입력 파라메터들(에러율(Error rate), 재연결율(Recovery rate), 태스크 수행 중지 확률(Stop rate), 로깅 오버헤드(Logging overhead), 호딩 오버헤드(Hoarding overhead), 재연결 오버헤드(Reintegration overhead)들이 비연결 수행 태스크 성능에 미치는 영향을 분석하였다. 이러한 분석을 통해서 통신망 단절을 고려한 이동컴퓨팅에서 보다 효과적인 태스크 수행기법을 선택할 수 있다.

ABSTRACT

Because wireless links are subject to disturbances and failures, it is important to support disconnected operations in mobile computing. Many schemes have been proposed to support the efficient disconnected operations. In this paper, we analyze and measure the computation time including disconnected operations to evaluate the performance of mobile computing on error prone wireless links. Mobile computation consists of three states: data hoarding, disconnected operation, and block states. We estimate the computation time using various parameters; error rate and recovery rate of wireless link, hoarding overhead, logging overhead, and reintegration overhead, etc. We can choose efficient strategies for disconnected operations and predict the performance using the results of this analysis.

Ⅰ. 서 론

무선네트워의 잦은 끊김과 높은 에러율로 인하여 이동 컴퓨팅에서는 비연결 수행에 관한 해결책을 제공해야 한다. 비연결 수행을 위한 태스크의 상태는 [그림1]과 같이 호딩(Hoarding), 비연결(Disconnected), 블록(Block) 상태로 구분된다. 프로세스(또는 데이터)는 비연결 상태에서도 수행이 가능하도록 이동 호스트쪽으로 데이터를 축적해야 한다. 이러한 비연결 수행상태 이전의 정상 상태를 호딩 상태(Hoarding State[1])라 한다. 무선네트워에 장애가 발생하면 이동 호스트는 비연결 상태(Disconnected state)가 된다. 이 상태에서도 이동 호스트는 이전의 호딩 상태에서 이동 호스트에 축적된 데이터를 이용하여 태스크의 수행을 계속 할 수 있다. 이러한 비연결 상태에서 수행 도중에 수정된 데

* 아주대학교 정보통신전문대학원 정보통신공학과 ({blueogre, jaikim}@ajou.ac.kr).
논문번호 : 010248-0917, 접수일자 : 2001년 9월 17일
※본 연구는 정보통신부에서 지원하는 대학기초연구지원 사업(2001-103-3)으로 수행되었습니다.

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

이터는 이동 호스트내에 저장해 두고 재연결시에 복구를 위해서 준비해 둔다. 재연결이 이루어지기 전에 이동 호스트(로컬)내에서 축적된 것 이외의 데이터를 요구하면 이동 호스트는 더 이상 태스크 수행을 진행하지 못하고 블록 상태(Block state)로 변한다. 블록 상태는 끊어진 무선 네트워크의 연결이 복구될 때까지 더 이상 수행을 할 수 없고 무선 연결이 복구될 때까지 대기하고 있어야 한다. 무선 네트웍이 재연결되면 비연결 상태에서 수행한 결과 및 사용했던 데이터 중에서 수정된 사항들을 기록한 이력(log)들을 다른 연결된 이동 호스트 또는 서버에게 알리게 된다. 이 때 재연결 오버헤드(Re-integration overhead)가 발생한다[1]. 이러한 재연결 수행은 이동 호스트가 비연결 상태에서 수행된 시간과 수정된 데이터의 양에 따라 오버헤드가 늘어나게 된다.

비연결 수행을 위한 많은 세부적인 기법들이 발표되었고 이들의 성능은 이동컴퓨팅 시스템 및 어플리케이션 특성에 따라 많은 차이를 보인다. 비안정적인 무선 환경에서 효과적인 비연결 수행 기법을 선택하기 위하여 시스템 환경 및 어플리케이션에 다른 비연결 수행 기법의 성능의 비교가 필요하다. 본 논문에서는 상태 전이도([그림1])를 통하여 비연결 수행비용을 분석하여 어플리케이션 프로그래머 또는 사용자가 적절한 비연결 수행 기법을 선택할 수 있도록 한다. 이러한 연구의 결과는 최근에 사용범위가 확대되고 있는 이동단말기를 이용한 이동컴퓨팅 응용분야에서 안정적인 무선통신망을 고려하여 계속적인 서비스를 효과적으로 제공하는데 이용될 수 있다.

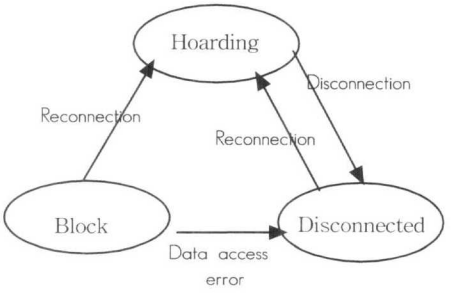


그림 1. 이동컴퓨팅의 3가지 상태

Ⅱ. 관련 연구

비연결 상태에서 태스크 수행을 효과적으로 지원하여 무선 이동 환경에서 응용 어플리케이션의 효율적인 실행을 위해서 많은 기법들이 제안되었다[7]. 파일 시스템의 비연결 수행을 지원하기 위해서 제안된 알고리즘의 대부분은 캐쉬기법을 기반으로 하고 있는데, 앞으로 이동 호스트에서 사용하게 될 내용을 미리 예측하여 무선 네트워크가 연결된 상태에서 이동 호스트 내의 기억 장소에 저장하는 방식이다. 무선 네트워크의 장애로 인해서 연결이 끊어진 상태에서도 이동 호스트 내에 저장된 데이터를 이용하여 태스크의 계속적인 수행을 가능하게 해준다. 이와 같이 네트워크가 연결된 상태에서 앞으로 사용하게 될 데이터를 미리 이동 호스트의 저장 공간에 저장하는 것을 호딩(hoarding), preloading, prefetch 등의 용어로 표현한다. 이와 같은 호딩 방식에는 과거의 패턴을 분석하여 앞으로 사용하게 될 파일을 예측하고 무선 네트워크가 연결된 상태에서 자동으로 축적하는 방법과 사용자가 직접 앞으로 사용하게 될 내용을 지정하여 축적하는 방법 중에 선택이 가능하다[1,2,3,4,5].

앞으로 사용하게 될 내용을 이동 호스트 내에 저장해 놓으면 무선환경이 단절되어 이동 호스트의 연결이 끊어지더라도 제한적이기는 하지만 태스크의 수행을 지속할 수 있다. 앞으로 사용될 내용을 예측하는 기법에는 두 가지 방법이 있는데 사용자의 예측이나 정해진 스케줄에 따르는 방법과 시스템에서 자동적으로 이전까지의 사용자의 사용 패턴을 기억하고 분석하여 그에 따라 적응하는 방법이 있다. CODA[2]와 FICUS[16]에서는 위에서 언급한 두 가지 방식을 혼합하여 사용하는 방식을 취하고 있다. 끊김이 발생하면 무선 이동 호스트는 자신에게 저장된 데이터를 사용하게 된다. 수행 도중에 저장된 데이터 이외의 데이터가 필요하게 되면 더 이상의 수행을 지속할 수 없으므로 수행은 정지하고 대기 상태로 네트워크가 복구되기만을 기다린다. 이러한 상태가 정지(STOP)상태이다. 이동 호스트는 비연결 상태에서 수행했던 내용에 대한 것을 로깅 파일(logging file)에 저장해 놓고 연결이 복구되면 저장한 로깅 파일을 이용해서 재연결 후에 데이터 일치성 유지에 관련된 내용을 수행한다. CODA에서는 위와 같은 데이터 조정을 위한 서버를 독립적으로 운영하는 모델을 사용하며, FICUS에서는 각각의 이동 호스트들 간에 직접 위와 같은 수행을 한다[12].

네트워크 연결성이 취약하고 대역폭이 낮은 무선 네트워크 환경에서도 파일 시스템과 유사하게 다양한 캐싱 기법을 적용하였다[10,11]. [5]에서는 데이터

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

베이스 시스템에서 적용된 예를 보여주고 있는데, 해당 시스템 사용자는 객체 지향 쿼리를 사용하며, 보관된 기존의 쿼리 패턴을 이용해서 쿼리를 만들고 객체 레코드를 유지한다. [6]에서는 호딩 쿼리를 위해서 데이터 베이스 관리자가 호드키(hoard key)라는 데이터 접근 패턴에 관한 내용을 테이블에 할당을 하여 데이터 베이스 시스템에서 효과적인 비연결 수행을 가능하게 하였다.

호딩 모델은 연결성이 취약하면서 낮은 대역폭의 무선 이동 환경에서 더 좋은 성능을 내고자 제안된 방식이다[8, 9]. [8]에서는 무선환경 뿐만 아니라 낮은 대역폭의 네트워크 환경에서는 호딩 전략을 사용하는 것이 유리하다는 것을 보이고 있다. [9]에서는 문맥인식(context- aware)정보 시스템에서 정보 갱신의 지연시간을 줄이기 위해서 호스트의 이동성(mobility)과 prefetching기법 등을 사용하였다. [17]에서는 데이터 일치성 유지를 위한 유연한 데이터 저장 시스템 모델을 다루고 있으며, [14]에서는 호딩을 위해서 무선 환경에서 지역성을 고려한 호딩의 성능 분석 및 내용에 대해서 다루고 있다

Ⅲ. 성능 분석

1. 호딩 기법의 성능 분석

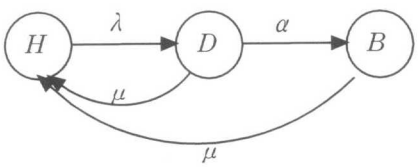


그림 2. 태스크의 상태 전이도

[그림 2]는 이동환경에서 태스크의 상태전이를 표시한 것이다. H는 Hoarding 상태, D는 Disconnected 상태이며 B는 Block 상태이다. 이때 λ는 연결이 끊어질 비율(rate)을 나타내며, μ는 끊어진 연결이 다시 재연결 될 비율을 나타낸다. 그리고 α는 연결이 끊어진 상태에서 수행을 지속하다가 재연결이 되지 않고 정지 상태로 이전할 비율을 나타낸다. 각각의 끊어짐과 재연결 그리고 블록은 poisson process를 따른다고 가정하였다. 이와 관련된 파라메터들을 정리하면 아래와 같다.

λ : 무선연결에 에러가 발생할 비율(rate)

μ : 끊어진 무선연결이 복원될 비율(rate)

α : 무선연결이 끊어진 상태에서 비연결 수행을 하는 도중에 BLOCK 상태로 전이할 비율(rate)

h ($h \geq 1$): 호딩 오버헤드율 (비연결 수행을 위해서 H상태에서 미리 데이터를 이동 호스트로 가져다 놓는데 필요한 오버헤드로 이 오버헤드에는 로깅 오버헤드, 재연결 오버헤드, 블럭킹 오버헤드는 포함되어 있지 않다. t시간의 유효 컴퓨팅(useful computation)을 위하여 ht시간이 필요하다.)

l ($l \geq 1$): 로깅 오버헤드율(로깅 오버헤드는 비연결 수행 시간 중 유효 컴퓨팅 시간에 비례한다고 가정하였다. 추가적으로 $(l-1)t$ 만큼의 오버헤드가 필요한데 이것은 비연결 수행시간 동안에 수정된 이동 호스트내의 데이터 이력을 저장하는데 소요되는 시간이다.)

c : 재연결 오버헤드율 (재연결 오버헤드는 데이터 로깅시간에 비례한다고 가정하였다. $c(l-1)t$ 만큼의 추가 비용이 요구되어지는데 이것은 lt 만큼의 비연결 수행이력을 다른 이동 호스트로 전파하는데 소요되는 시간이다.)

[그림 2]를 이용하여 태스크를 수행하는데 요구되는 비용을 분석하였다.

P_H, P_D, P_B를 [그림 2]에서 표현된 H, D, B의 상태에 있을 확률이라고 정의하면 아래와 같은 식을 유도할 수 있다.

① $P_H + P_D + P_B = 1$

② $-\lambda P_H + \mu P_D + \mu P_B = 0$

③ $\lambda P_H - (\alpha + \mu) P_D = 0$

④ $\alpha P_D - \mu P_B = 0$

Using the equations , we can compute values P_H, P_D, and P_B as follows:

위의 ①~④ 식을 이용하여 P_H, P_D, P_B값을 구하면 아래와 같다.

$$P_H = \frac{\mu}{\lambda + \mu}$$

$$P_D = \frac{\lambda\mu}{(\alpha + \mu)(\lambda + \mu)}$$

$$P_B = \frac{\alpha\lambda}{(\alpha + \mu)(\lambda + \mu)}$$

$f(t)$를 시간 t 만큼의 유효컴퓨팅(useful computation)을 진행하기 위한 시간이라고 하면

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

$f(t+\varepsilon)-f(t)$ 는 시간 t 로부터 ε만큼의 유효 컴퓨팅 시간을 수행하는데 소요되는 시간이 된다. 시간 t 로부터 ε만큼의 간격을 수행하는 시간을 위의 [그림 2]에 적용하여 구하면 크게 (1) H상태에서 시작하는 경우, (2) D상태에서 시간 계산을 시작하는 경우의 두 가지로 나누어진다. 이러한 각각의 상태로부터 ε간격을 수행한 비용 $f(t+\varepsilon)-f(t)$ 을 $H(t,\varepsilon)$과 $D(t,\varepsilon)$라고 하고 아래와 같이 계산하였다.

(1) $H(t,\varepsilon)$ 계산하는데 다음과 같은 경우들로 나누어 계산하였다.

- ε 간격 사이에 끊김이 발생하지 않고 H상태에서만 수행되는 경우($e^{-\lambda\varepsilon}$) : ε간격동안 ht 만큼의 시간이 요구되어 진다.
- ε 간격 사이에 끊김이 발생하여 H상태에서 D상태로 전이되는 경우($1-e^{-\lambda\varepsilon}$) : $E_\lambda(\varepsilon)$은 ε간격동안 수행하는 도중에 연결이 λ의 확률을 가지고 끊김이 발생한다고 가정할 때, 끊김이 발생한 경우 그전에 수행한 평균시간이다. 평균시간은 아래와 같은 식으로 구할 수 있다.

$$E_\lambda(\varepsilon)=\int_0^\varepsilon t\frac{\lambda e^{-\lambda t}}{1-e^{-\lambda t}}dt = \lambda^{-1}-\frac{\varepsilon e^{-\lambda\varepsilon}}{1-e^{-\lambda\varepsilon}}$$

끊김이 발생하기 전에 H상태에서 $E_\lambda(\varepsilon)h$ 만큼의 시간이 필요하고 연결이 끊어진 후에는 $\frac{\varepsilon-E_\lambda(\varepsilon)}{\varepsilon}D(t,\varepsilon)$ 만큼의 시간이 필요하다.

따라서 위의 식에서 $H(t,\varepsilon)$를 구하면 다음과 같다.

$$H(t,\varepsilon) = f(t+\varepsilon) - f(t)$$
$$= e^{-\lambda\varepsilon}h\varepsilon+(1-e^{-\lambda\varepsilon})\left[E_\lambda(\varepsilon)h+\frac{\varepsilon-E_\lambda(\varepsilon)}{\varepsilon}D(t,\varepsilon)\right]$$

(2) D 상태에서 시작하게 되는 $D(t,\varepsilon)$는 3가지 경우로 나누어 생각할 수 있다.

- ε 동안 계속 D상태에서 재연결이나 blocking 없이 수행되는 경우($e^{-\mu\varepsilon}e^{-\alpha\varepsilon}$) : ε간격동안 $l\varepsilon$만큼의 시간이 요구되어 진다.
- ε 도중에 재연결이 발생하여 D상태에서 H상태로 전환되는 경우($1-e^{-\mu\varepsilon}$) : 재연결이 되기 전에 $lE_\mu(\varepsilon)$만큼의 시간이 필요하고 재연결 하는데 $c(l-1)(1/\mu)$ 의 시간이 필요로 하며, 재연결 후에 $\frac{\varepsilon-E_\mu(\varepsilon)}{\varepsilon}H(t,\varepsilon)$ 만큼의 시간이 요구된다.
- ε 도중에 blocking이 일어나 상태로 될 경우 ($1-e^{-\alpha\varepsilon}$) : block이 되기 전까지 $lE_\alpha(\varepsilon)$ 만큼의 시간이 요구되며, 평균 $\frac{1}{\mu}$ 만큼의 block된 시간이 소모된다. 그리고 다시 재연결 시에 $c(l-1)(1/\alpha)$ 만큼의 시간이 필요하며, 재연결 후에는 $\frac{\varepsilon-E_\alpha(\varepsilon)}{\varepsilon}H(t,\varepsilon)$ 의 시간이 요구된다.

따라서 위의 세 가지 경우에 대해서 합쳐서 정리를 하여 $D(t,\varepsilon)$를 구하면 아래와 같다.

$$D(t,\varepsilon) = f(t+\varepsilon)-f(t)$$
$$= e^{-\mu\varepsilon}e^{-\alpha\varepsilon}l\varepsilon$$
$$+(1-e^{-\mu\varepsilon})\left[lE_\mu(\varepsilon)+c(l-1)(\frac{1}{\mu})+\frac{\varepsilon-E_\mu(\varepsilon)}{\varepsilon}H(t,\varepsilon)\right]$$
$$+(1-e^{-\alpha\varepsilon})\left[lE_\alpha(\varepsilon)+\frac{1}{\mu}+c(l-1)(\frac{1}{\alpha})+\frac{\varepsilon-E_\alpha(\varepsilon)}{\varepsilon}H(t,\varepsilon)\right]$$

(1), (2)에 대해서 각각 ε에 대하여 미분을 계산하면 아래와 같은 값을 얻을 수 있다.

$$\lim_{\varepsilon\to0}\frac{H(t,\varepsilon)}{\varepsilon}$$
$$= \lim_{\varepsilon\to0}\left[h + E_\lambda(\varepsilon)h + \lambda\frac{\varepsilon-E_\lambda(\varepsilon)}{\varepsilon}D(t,\varepsilon)\right]$$
$$= h$$

$$\lim_{\varepsilon\to0}\frac{D(t,\varepsilon)}{\varepsilon}$$
$$= \lim_{\varepsilon\to0}\left[l+\mu\left\{lE_\mu(\varepsilon)+c(l-1)(\frac{1}{\mu})+\frac{\varepsilon-E_\mu(\varepsilon)}{\varepsilon}H(t,\varepsilon)\right\}\right.$$
$$\left.+\alpha\left\{lE_\alpha(\varepsilon)+\frac{1}{\mu}+c(l-1)(\frac{1}{\alpha})+\frac{\varepsilon-E_\alpha(\varepsilon)}{\varepsilon}H(t,\varepsilon)\right\}\right]$$
$$= l + c(l-1) + \frac{\alpha}{\mu}+c(l-1)$$
$$= l + \frac{\alpha}{\mu}+2c(l-1)$$

$g(t)$를 안정상태(Steady State)에서 시간 t만큼의 유효컴퓨팅(useful computing)을 수행하는데 필요로 하는 기대시간이라고 하면 ε 간격동안의 비용 $g(t+\varepsilon)-g(t)$ 을 계산하기 위하여 아래와 같이 표현한다.

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

$$g(t+\varepsilon)-g(t) = \frac{P_H H(t,\varepsilon)+P_D D(t,\varepsilon)}{P_H+P_D}$$

위 식을 ε에 대하여 미분하면 다음과 같은 식을 얻을 수 있다.

$$\lim_{\varepsilon\to 0}\frac{g(t+\varepsilon)-g(t)}{\varepsilon}$$

$$= \frac{1}{P_H+P_D}\left[P_H \lim_{\varepsilon\to 0}\frac{H(t,\varepsilon)}{\varepsilon} + P_D \lim_{\varepsilon\to 0}\frac{D(t,\varepsilon)}{\varepsilon}\right]$$

$$\Rightarrow \frac{\partial g(t)}{\partial t} = \frac{1}{P_H+P_D}\left[P_H h + P_D\left\{l+\frac{\alpha}{\mu}+2c(l-1)\right\}\right]$$

$$= \frac{(\alpha+\mu)(\lambda+\mu)}{(\alpha+\lambda+\mu)\mu}\left[\frac{(\alpha+\mu)\mu h+\lambda\mu\left\{l+\frac{\alpha}{\mu}+2c(l-1)\right\}}{(\alpha+\mu)(\lambda+\mu)}\right]$$

$$= \frac{(\alpha+\mu)\mu h+\lambda(\mu l+\alpha)+2\lambda\mu c(l-1)}{(\alpha+\lambda+\mu)\mu}$$

$$\Rightarrow g(t) = \frac{(\alpha+\mu)\mu h+\lambda(\mu l+\alpha)+2\lambda\mu c(l-1)}{(\alpha+\lambda+\mu)\mu}t + k$$

위의 식에서 $g(0)=0$ 이므로 $k=0$ 이 된다. 따라서

$$g(t) = \frac{(\alpha+\mu)\mu h+\lambda(\mu l+\alpha)+2\lambda\mu c(l-1)}{(\alpha+\lambda+\mu)\mu}t \quad \cdots \quad (A)$$

와 같은 식을 얻을 수 있다. 위 식을 이용해서 각 환경 변수들의 값에 따른 비연결 수행 비용을 산출해 낼 수 있다.

2. 호딩 기법의 선택 여부

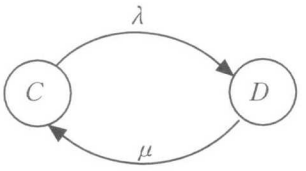


그림 3. 호딩이 없는 비연결 태스크의 수행 상태전이도

특정한 환경 하에서는 비연결 수행을 위해서 호딩 및 이에 다른 부가적인 기능들이 오버헤드가 되어 오히려 호딩 수행을 하지 않은 편이 더 좋은 성능을 낼 수도 있기 때문에 [그림3]과 같은 모델을 만들어서 호딩 기법을 사용하지 않을 경우의 수행 시간을 분석하였다. [그림 3]은 이러한 이동 환경에서 태스크의 상태 전이도를 나타낸 것이다. C 상태는 연결이 되어 태스크를 수행하는 상태이고 D 상태는 연결이 끊어진 상태로 태스크 수행을 멈춘 disconnect 상태이다.

호딩을 포함한 방식을 계산했던 것과 마찬가지로 $g(t)$를 얻기 위해서 다음과 같이 계산을 수행한다. 먼저 안정상태에서 각 상태에 존재할 확률을 구하면,

① $P_C + P_D = 1$

② $-\lambda P_C + \mu P_D = 0$

$\Rightarrow P_C = \frac{\mu}{\mu+\lambda}$, $P_D = \frac{\lambda}{\mu+\lambda}$ 의 값을 얻는다

호딩 데이터가 없으므로 C 상태에서만 태스크의 수행을 하게 되므로 D 상태에서 수행되는 유효컴퓨팅(useful computation)은 없다. 따라서 시간 t로부터 ε 의 유효컴퓨팅(useful computation)에 소요되는 시간 $C(t,\varepsilon)$ 을 구하면 전체 수행 시간을 얻을 수 있다.

(1) $C(t,\varepsilon)$

- ε 동안 진행하는 동안 끊어짐이 없이 C 상태에서 수행되는 경우($e^{-\lambda\varepsilon}$) : ε 동안 진행하는 동안 ε 만큼의 시간이 요구되어 진다(호딩 오버헤드 없음).
- ε 동안 진행하는 동안 연결이 끊어져서 D 상태로 전환되는 경우($1-e^{-\lambda\varepsilon}$) : 평균 $\frac{1}{\mu}$ 시간 동안의 재연결시까지 대기시간을 가진다.

따라서 위의 두 가지 경우에서 $C(t,\varepsilon)$를 계산해 보면

$$C(\varepsilon,t) = e^{-\varepsilon\lambda}\varepsilon + (1-e^{-\varepsilon\lambda})(\varepsilon+\frac{1}{\mu})$$

이며 이 식을 ε 에 대하여 미분을 하면 아래와 같다.

$$\lim_{\varepsilon\to 0}\frac{e^{-\varepsilon\lambda}\varepsilon+(1-e^{-\varepsilon\lambda})(\varepsilon+\frac{1}{\mu})}{\varepsilon} = 1+\frac{\lambda}{\mu}$$

따라서 호딩이 없는 비연결 수행에서의 t 시간만큼 useful computation을 위하여 필요한 시간은 아래와 같다.

$$g(t) = \left(\frac{\lambda}{\mu}+1\right)t \quad \cdots\cdots \quad (B)$$

결론적으로 식 (A)와 (B)를 이용하여 주어진 환경 변수를 사용해서 평균 수행시간을 비교하면 현재의 무선 이동 환경이 호딩 알고리즘을 사용하는 것이 유리한지 아니면 이동 환경임에도 불구하고 호딩 알고리즘을 사용하지 않는 것이 유리한지 비교가 가능하다. 아래와 같은 조건을 만족하는 경우

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

호딩 알고리즘을 적용하여 사용하는 것이 이동 환경에서 더 좋은 성능을 기대할 수 있다. 그렇지 않은 경우에는 호딩을 사용하는 것이 오히려 오버헤드가 늘어나서 역효과를 얻을 수 있다.

$$\frac{(\alpha+\mu)\mu h+\lambda(\mu l+\alpha)+2\lambda\mu c(l-1)}{(\alpha+\lambda+\mu)\mu} < (\frac{\lambda}{\mu}+1)$$

$$\frac{(\alpha+\mu)\mu h+\lambda(\mu l+\alpha)+2\lambda\mu c(l-1)}{(\alpha+\lambda+\mu)(\mu+\lambda)} < 1 \quad \cdots \text{(C)}$$

위의 식(C)을 이용해서 환경에 따라 적응적인 이동 알고리즘을 적용할 수 있다. 즉, 이동컴퓨팅 시스템의 파라메터 및 어플리케이션의 특성에 따라 서로 다른 무선 통신망의 끊김 에러율(λ)과 재연결율(μ), 끊어진 상태에서 정지 상태로 갈 확률(α), 그리고 비연결 수행을 함으로써 발생하는 호딩, 로깅, 및 재연결 오버헤드 등의 인자들을 바탕으로 적절한 비연결 수행기법을 선택할 수 있고 경우에 따라 비연결 수행여부도 결정할 수 있다.

[그림5]부터 [그림8]에서는 위의 식을 이용하여 호딩이 유리한 경우와 불리한 경우를 구분하였다. 색이 칠해진 부분에서는 호딩 기법이 불리한 경우를 나타내며 색이 없는 부분은 호딩 기법이 유리한 경우를 나타낸다. [그림5]와 [그림6]에서는 λ 값이 클수록, μ 값이 작을수록, α 값이 작을수록 호딩 알고리즘을 사용하는 것이 유리하다. [그림8]은 λ 값이 작을 때는 h 값의 변화에 덜 민감하지만 λ 값이 커질수록 h 값의 변화에 영향을 많이 받는 것을 알 수 있다.

Ⅳ. 성능 비교

위에서 정의한 이동 환경에 영향을 주는 환경 변수들에 따른 이동 컴퓨팅의 수행 성능을 비교하였다. 비연결 수행 오버헤드를 computation time과 유효컴퓨팅(useful computation)과의 비율이라고 정의하자. computation time에는 로깅 오버헤드, 재연결 오버헤드, 블록킹 오버헤드 등 각종 비연결 수행에 필요한 오버헤드 들이 포함되어 있다. 유효컴퓨팅(useful computation)은 computation time과는 다르게 위와 같은 비연결 수행을 위한 오버헤드들을 제외하고 오직 태스크의 수행 시간만을 나타낸다. [그림8]부터 [그림14]까지에서 사용되는 문자 l, m, a, h, L, c 는 각각 λ, μ, α, h, l, c 에 대응된다. 그리고 [그림4]에서 [그림8]까지 표현된 점선은 호딩이 없는 상태에서의 결과를 나타낸다.

[그림4]는 α 값에 변화를 주면서 λ값 을 증가시킨 결과이다. 결과는 예상대로, λ, α 값이 커질수록 높은 오버헤드를 보이고 있다. 호딩이 없는 경우 역시 유사한 모양의 성능을 보이면서 α 값이 높을 때와 비슷한 성능을 보여주었다. 이는 α 값이 커질수록 비연결 수행에서 얻는 이득보다는 BLOCK 상태에 있는 시간이 길어지기 때문이다. [그림5]와 [그림6]에서는 h 값에 변화를 주면서 λ 값을 증가시켰다. 결과는 λ 값이 작을 때는 h 값의 변화는

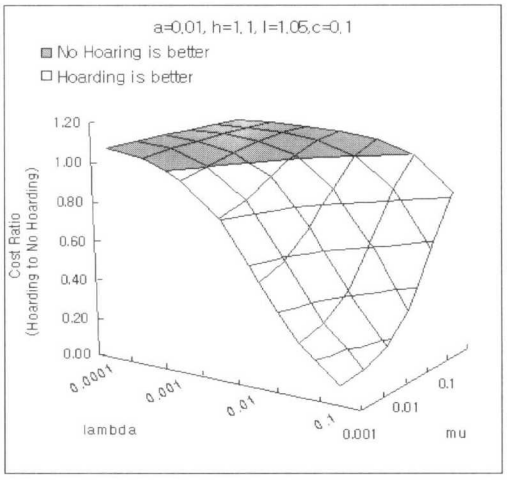


그림 4. λ 값과 μ 값의 변화에 따른 호딩과 비호딩의 비용 비교

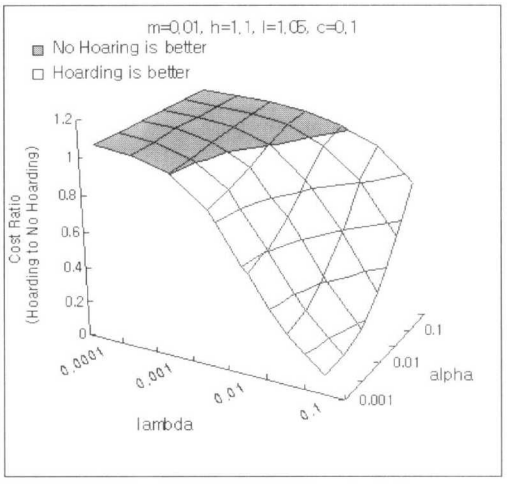


그림 5. λ 값과 α 값의 변화에 따른 호딩과 비호딩의 비용 비교

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

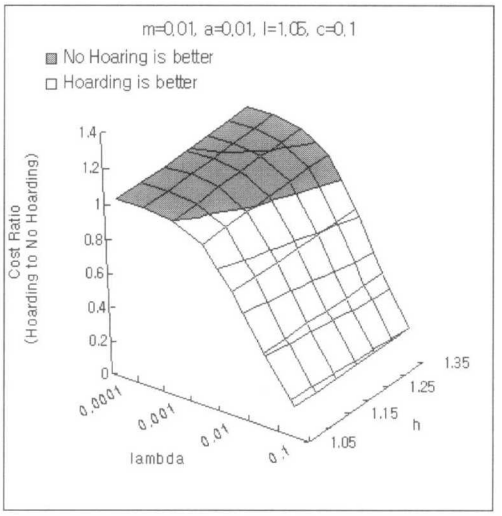


그림 6. λ 값과 h 값의 변화에 따른 호딩과 비호딩의 비용 비교

성능에 많은 영향을 주었다. 반대로 λ 값이 커지면서 상대적으로 h 값의 변화에 영향을 덜 받는 결과를 보여준다. 이것은 λ 값이 작을 때는 H 상태를 오랫동안 유지하기 때문이다. 여기서 주목할 것은 h 값에 따른 그래프의 변화이다. 값이 1.2 이상일 경우에는 λ 값이 0.01을 넘으면서 점차 감소하게 되었지만 h 값이 1.2 이하일 경우에는 서서히 증가되는 것을 볼 수 있다. h 값이 클 때 l 값이 상대적으로 적다면 H 상태에 있는 것 보다 D 상태에 있는 것이 유리하지만, h 값이 적다면 H 상태에 있는 것이 유리하기 때문이다. 또한 [그림5]와 [그림6]은 로깅 오버헤드의 값이 서로 다른데 그래프에서 보이듯이 λ 값이 작을 때는 로깅 오버헤드의 값에 따라 성능에 별 차이가 없지만 λ 값이 커지면서 로깅 오버헤드의 값의 차이가 성능에 영향을 많이 주는 결과를 보인다. 이는 λ 값이 커지면 D 상태에 있을 가능성이 높아지고 이때 l 값이 크면 로깅 부담이 커지기 때문이다. [그림7]에서는 로깅 오버헤드의 값에 변화를 주면서 λ 값을 증가시켰다. 위에서 설명한 바와 마찬가지로 로깅 오버헤드는 D 상태에 머무르는 시간에 비례하게 되므로 λ 값이 커질수록 D 상태에 존재할 가능성이 높아져 l 값에 많은 영향을 받는다. [그림8]에서는 c 값에 변화를 주면서 λ 값을 증가시켰다. 이 결과 마찬가지로 λ 값이 작을 때는 재연결 오버헤드 값이 작기 때문에 근소한 차이가 나지만 λ 값이 커질수록 그 차이는 커지게 된다. 또한 [그림5]에서 [그림11]에 표현된 점선은 호딩 알고리즘을 적용하지 않은 경우의 성능을 표현한 것이다. [그림12], [그림13]에서는 μ 값을 변화하면서 λ 값을 증가시켰다. [그림12]에서 λ 값이 증가하면서 낮은 μ 값 ($\mu \leq 0.15$)일 경우에는 낮은 성능을 보여주고 있지만 비교적 높은 μ 값($\mu \geq 0.15$) 일 경우에는 성능이 좋아지고 있다. 일반적으로 λ 값이 크면 성능이 낮아져야 하는데 μ 값이 클 때 오히려 좋은 성능을 보여주고 있다. 이유는 호딩 오버헤드가 클 경우 ([그림12]에서 h =1.25) 로깅 오버헤드와 블록 전이율이 비교적 작다면 ([그림12]에서 l =1.05, α =0.01) H 상태에 있는 것보다 D 상태에 있는 것이 유리하기 때문이다. 따라서 항시 연결 수행이 유리한 것이 아니라 경우에 따라 의도적으로 비연결 수행을 하는 것이 유리한 경우도 있다. [그림 14]는 비교를 위한 비호딩시의 그래프를 나타낸 것이다.

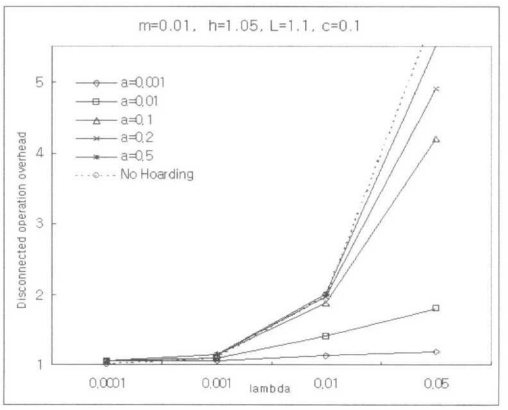


그림 7. α 변화에 따른 λ 값의 변화

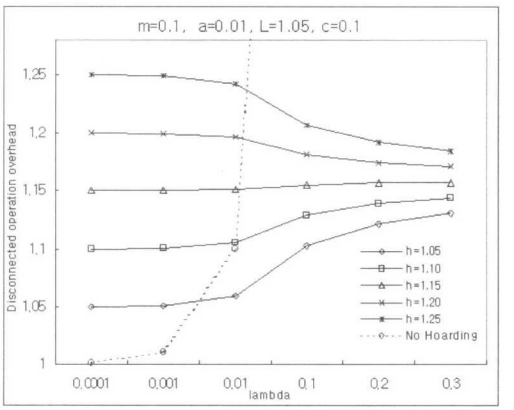


그림 8. h 변화에 따른 λ 값의 변화(l =1.05)

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

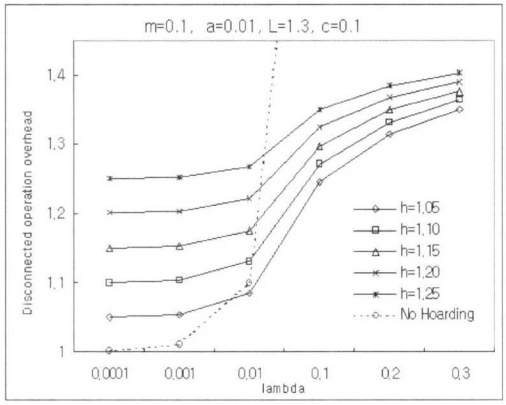


그림 9. h 변화에 따른 λ 값의 변화(l =1.3)

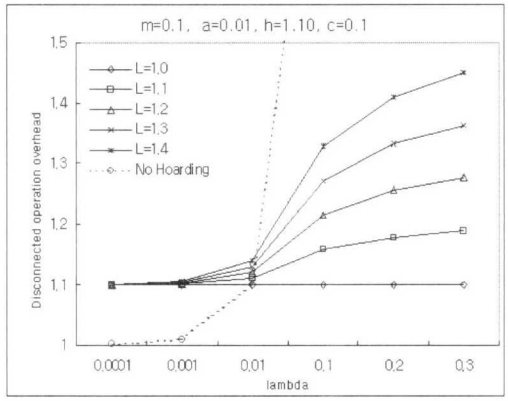


그림 10. l 변화에 따른 λ 값의 변화

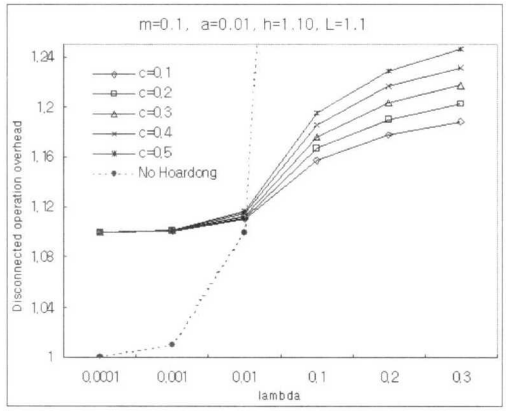


그림 11. c 변화에 따른 λ 값의 변화

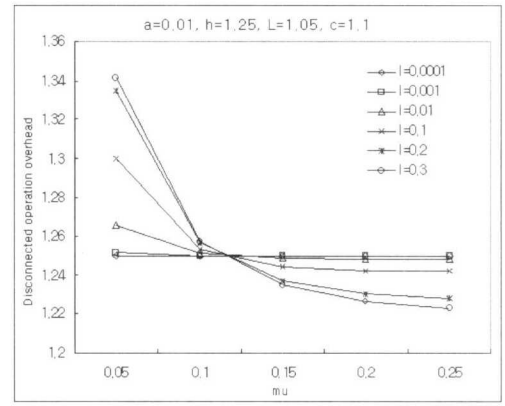


그림 12. μ 변화에 따른 λ 값의 변화

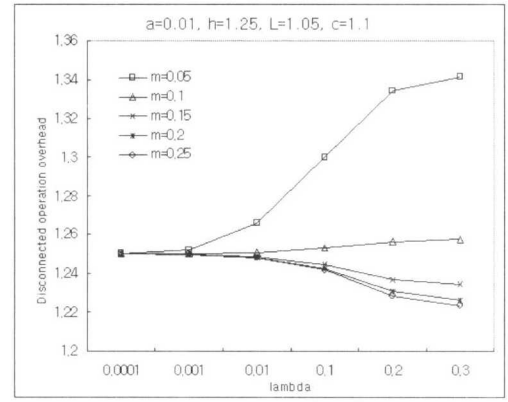


그림 13. λ 변화에 따른 μ 값의 변화

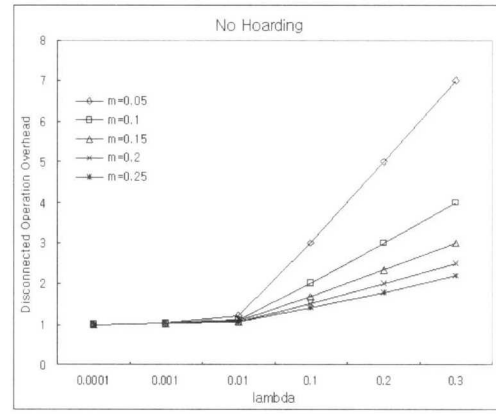


그림 14. No Hoarding

잦은 끊김과 잦은 재연결로 인해서 [그림2]에서 H, D 상태를 반복하면서 재연결 오버헤드가 증가하기 때문이다. 따라서 무선 환경에서 반드시 비연결 수행을 해야만 좋은 성능을 기대하는 것이 아니라 성능에 영향을 미치는 환경 요소들을 먼저 분석하고 비연결 수행을 사용하는 것이 보다 더 나은 성능을 기대 할 수 있을 때 비연결 수행을 하는 것이 유리하다.

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

V. 결 론

이동 환경에서의 비연결 수행에 영향을 주는 요소들을 분석하고 비연결 수행의 상태를 Hoarding, Disconnected, Block 상태로 분석하고 비연결 수행에 영향을 미치는 요소들을 분석하였다. 본 논문에서는 이와 같은 요소들을 다음과 같이 나누어서 비용을 분석하였다. 연결이 끊어질 비율, 끊어진 연결이 복구될 비율, 이동 컴퓨팅 환경에서 비연결 수행 도중에 blocking 될 비율, 호딩 오버헤드율, 로깅 오버헤드율, 재연결 오버헤드율 등이 있다.

비연결 수행기법을 사용하는 것은 무선통신 환경뿐만 아니라 기존 유선망에서도 이용이 가능하며 비연결 수행을 사용하면서 통신망의 트래픽 조절이 가능하므로 보다 효율적인 망 관리에 도움을 줄 수 있다. 그러나 본 논문에서 보인 바와 같이 비연결 수행에서 호딩 알고리즘을 적용하는 것이 반드시 좋은 성능을 기대할 수 있는 것은 아니다. 이동 환경에 따라 호딩 오버헤드율이 클 경우 호딩 기법을 사용하지 않고 재연결을 기다리는 것이 더 유리하다. 또한 로깅 오버헤드율과 블록 전이율이 작은 경우에 호딩 오버헤드율이 클 경우 무선 연결이 복구가 되더라도 고의적으로 비연결 수행 모드에서 수행을 지속하면서 호딩시에 저장된 데이터를 사용하여 호딩 오버헤드 비율을 줄이는 방식이 유리하다. 이러한 적응적인 비연결 수행 기법을 사용하는 것이 보다 높은 성능을 기대할 수 있으며, 본 논문에서 수행한 분석은 향후 이동 환경에서의 효과적인 비연결 수행 기법을 선택하는데 활용될 수 있다.

참 고 문 헌

[1] E. Pitoura and G. Samaras, "Data Management for Mobile Computing," *kluwer Academic Publishers*, pp. 37-48, 1997.

[2] J. J. Kistler and M. Satyanarayanan, "Disconnected Operations in the Coda File System," *ACM Transactions on Computer Systems*, 10(1), pp. 213-225, February 1992.

[3] C. Tait, H. Lei, S. Acharya, and H. Chang, "Intelligent File Hoarding for Mobile Computers," *In Proceedings of the 1st ACM International Conference on Mobile Computing and Networking*, October 1995.

[4] G. H. Kuenning, "The Design of the Seer Predictive Caching System," *In Proceedings of the IEEE Workshop on Mobile Computing system and Applications*, December 1994.

[5] R. Gruber, F. Kaashoek, N. Liskov, and L. Shrira, "Disconnected Operation in the Thor Object-Oriented Database System," *In Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, December 1994.

[6] I. Akyildiz and J. Ho, "Dynamic Mobile User Location Update for Wireless PCS Networks," *ACM/Baltzer Wireless Networks Journal*, 1(2), 1995.

[7] T. Ye, H. Jacobsen and R. Katz, "Mobile awareness in a area wireless network of info- stations," *In Proceedings of ACM International Conference on Mobile Computing and Networking*, October 1998.

[8] N. Tuah, M. Kumar, and S. Venkatesh, "Investigation of a Prefetch Model for Low Bandwidth Networks," *In Proceedings of International Workshop on Wireless Mobile Multimedia*, pp. 38-47, 1998.

[9] V. Persone, V. Grassi, and A. Morlupi, "Modeling and Evaluation of Prefetching Policies for Context-Aware Information Services," *In Proceedings of The International Conference on Mobile Computing and Networking*, pp. 55-65, 1998.

[10] Q. Li and D. Rus, "Sending Message to Mobile Users in Disconnected Ad-hoc Wireless Networks," *In Proceedings of The International Conference on Mobile Computing and Networking*, pp. 44-55, 2000.

[11] R. Kavasseri, T. Keating, and Michael Wwittman, "A. Joshi, S. Weerawarana, Web Intelligent Query - Disconnected Web Browsing Using Cooperative Techniques," *In Proceedings of the First IFCIS International Conference on Cooperative Information Systems*, pp. 167-174, 1996.

[12] Shirish Hemant Phatak and B. R. Badrinath, "Data Partitioning for Disconnected Client Server Databases," *MobiDE*, pp. 102-109, 1999.

[13] G. Kuenning and Gerald J. Popek, "Automated Hoarding for Mobile Computers," *In Proceedings Sixteen [ACM] Symposium on Operating Systems Principles*, Saint Malo, France, pp. 264-275, 1997.

[14] U. Kubach and K. Rothermel, "A Map-Based Hoarding Mechanism for Location-Dependent

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr

Information," *In Proceedings of the Second International Conference on Mobile Data Management (MDM 2001)*, Hong Kong, January 2001.

[15] B. Noble and M. Satyanarayanan. "A Research Status Report on Adaptation for Mobile Data Access," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 24(4), December 1995.

[16] G. Kuenning, G. J. Popek and P. Reiher, "An Analysis of Trace Data for Predictive File Caching in Mobile Computing," *In Proceedings of the USENIX Summer Conference*, pp. 191-303, 1994.

[17] N. Preguica, J. Legatheaux Martins, and H. J. Domingos, "Flexible Data Storage for Mobile Collaborative Applications," *In Proceedings of the Third European Research Seminar on Advances in Distributed Systems(ERSADS '99)*, April 1999.

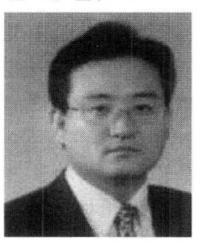
정 승 식(Seong-Sik Jung) 정회원

2000년 2월 : 아주대학교 컴퓨터 공학과 (학사)

2002년 2월 : 아주대학교 정보통신전문대학원 정보통신 공학과 (석사)

<주관심 분야> Distributed Systems, Real-Time Systems, Embedded system, Mobile Computing

김 재 훈(Jai-Hoon Kim) 정회원



1984년 : 서울대학교 제어계측학과 (학사)

1993년 : 미국 Indiana 대학 컴퓨터 공학(석사)

1997년 : 미국 Texas A&M 대학 컴퓨터 공학(박사)

1984～1991 : 대우통신 종합연구소, 컴퓨터연구실 팀장

1988년 Tolerant Systems 참여 연구원

1995～1997 : Graduate Assistant Research Department of Computer Science, Texas A & M University.

1997～1998 : 삼성전자 컴퓨터시스템 개발팀, 수석 연구원

1998～ : 아주대학교 정보통신전문대학원 조교수

<주관심 분야> Distributed Systems, Mobile Computing, Real-Time Systems, Fault-Tolerant System

Copyright (C) 2003 NuriMedia Co., Ltd.
www.dbpia.co.kr