韓國컴퓨터情報學會 論文誌
第15卷 第1號, 2010. 1.

2010-15-1-1-8

분산된 환경에서의 상태 전이 그래프의 확장

서 진 형*, 이 왕 헌**

An extension of state transition graph for distributed environment

Jin Hyung Suh*, Wang Heon Lee**

요 약

일반적인 웹 환경에서도 웹 페이지에 포함되어 있는 웹뷰 콘텐츠의 갱신 상태와 재-계산 상태 또는 생성되는 과정에서의 변화는 파악하기가 어려우며, 만약 갱신이 되고 있는 도중에 읽기 작업이 수행되거나, 잘못된 갱신으로 인한 부정확한 콘텐츠의 사용으로 인한 문제가 발생하면 이에 대한 문제 해결은 매우 복잡하다. 이러한 문제를 해결하기 위하여 많은 연구가 이루어져 왔으나, 대부분 단일 사용자 환경에서의 웹뷰 연구로 분산된 환경에서의 적용은 문제가 있다. 이러한 이유로 본 연구에서는 웹뷰 콘텐츠의 운영 상태를 확인할 수 있는 방법으로 기존의 상태 전이 그래프를 확장하고 이에 대한 알고리즘을 제시하여 분산된 환경에서의 웹뷰 콘텐츠의 갱신 및 재-계산 상태 또는 생성되는 과정에서의 상태 변화를 설명할 수 있도록 하며, 구체화 웹뷰를 이용한 실험을 통하여 구체화 웹뷰를 사용하였을 경우와 사용하지 않았을 경우 그리고 네트워크의 상태에 따른 시간 분석과 웹뷰 콘텐츠의 크기에 따른 효율성에 대한 분석을 한다.

Abstract

In a typical web environment, it is difficult to determine the update and re-computation status of WebView content or the transition of WebView processing included in web page. If an update to one of data is performed before a read operation to that, we could get a wrong result due to the incorrect operation and increase a complexity of the problem to process. To solve this problem, lots of researchers have studied and most of these problems at the single user environment is not problems. However, the problems at a distributed environment might be occurred. For this reason, in this paper, we proposed the extended state transition graph and algorithms for each status of WebView for explaining WebView state in the distributed environment and analyze the performance of using the materialized WebView and not. Additionally, also analyze the timing issues in network and effectiveness which follows in size of WebView contents.

▸ Keyword : 웹뷰 (WebView), 상태전이그래프 (state transition graph), 분산 환경 (distributed environment)

• 제1저자 : 서진형 교신저자 : 이왕헌
• 투고일 : 2009. 12. 18, 심사일 : 2009. 12. 23, 게재확정일 : 2010. 01. 26.
* 경인여자대학 정보미디어과 교수 ** 한세대학교 IT 학부 교수

Ⅰ. 서 론

일반적으로 웹 캐싱 (web caching) [1]과는 달리 웹뷰(Web View)는 사용자와 웹/응용 서버 그리고 데이터베이스 서버로 구성된 3 단계 구조 (3-tier architecture) 또는 3 단계 구조에서 웹과 응용 서버를 분리한 4 단계 구조 (4-tier architecture) [2]를 기본으로 하며, 이를 통하여 사용자가 요구하는 정보를 구체화시켜 제공하게 된다. 그러나 현재의 웹 환경에서 사용자가 요구하는 정보는 여러 장소에 분산 저장되고 있으며, 이들 정보를 이용하여 사용자가 요구하는 결과물을 병합하여 웹 페이지를 생성하게 된다. 즉, 웹뷰 콘텐츠는 웹 페이지를 구성하는 부분품 (fragment)으로, 이를 기반으로 하여 사용자가 원하는 웹 페이지를 구성하게 된다. 그러나 사용자는 웹 페이지가 구성되는 동안 웹뷰 콘텐츠의 사용 및 구성 상태의 확인뿐만 아니라 웹뷰의 생성과 폐기라는 개념이 적용되어 있지 않아 언제 만들어지고 사용되는지를 알 수 없다. 이를 해결하기 위하여 상태 전이 그래프 [3]가 제시되었으나 이 역시 지역적으로 분산되어 있는 환경이 아닌 단일 사용자 환경에서의 연구이다.

Ⅱ. 관련 연구

1. 관련연구

현재 대부분의 웹 환경은 단일 사용자 환경에서 분산 사용자 환경으로 변화하고 있는 상태로 이에 대한 많은 연구가 이루어지고 있다. 그러나 분산 환경은 적절히 분산되어 있는 각 웹 서버의 정보를 다양하게 활용할 수 있는 환경을 제공하여야 사용자가 요구하는 정보를 얻을 수 있으며, 이를 위한 적합한 데이터 저장 및 웹뷰 유지 방법이 제시되어야 한다. 그러나 일반적인 구조를 적절히 분산되어 있는 환경에 적용할 경우 사용자의 질의에 해당하는 정보를 추출할 수 있는 방법, 네트워크 연결시의 지연과 네트워크의 장애등으로 인하여 많은 시간적인 문제와 상대방에서의 잘못된 정보의 전달 등으로 인하여 사용자가 요구하는 정보를 전달할 수 없다.

현재 일반적인 웹뷰에 대한 연구로는 [6]과 분산된 환경에서의 연구로는 [4,5]가 있는데 이 연구에서 분산 환경에서의 웹뷰 구체화 방법 [6]과 구체화 모델에 대하여 연구하였다. 이에 대한 결과로 분산된 네트워크 환경에서의 웹뷰 구체화를 위한 확장된 조건을 검색하여 보면 실제 일반적인 분산 시스템에서의 해결하여야 하는 문제를 확대한 것으로 파악할 수 있으며, 추가로 기존의 3 단계 모델과 이에 적합한 질의어의 확장 역시 요구됨을 확인할 수 있으나, 이외의 연구는 국내에서는 미진한 상태이다.

[7]의 연구에서는 단일 사용자 환경에서 여러 개의 데이터베이스에 저장되어 있는 데이터 집중 웹 사이트 (data-intensive web site; DIWS)에서의 웹뷰 모델을 제시하여 지역적으로 분산되어 있는 웹 사이트에서의 웹뷰 구체화 방법과 운영에 대한 정의를 하였으나 분산된 환경에서의 웹뷰 구체화에는 적합하지 못하다.

[3]의 연구에서의 상태 전이 그래프는 단일 사용자 환경 또는 정적인 사용 환경에서의 웹뷰의 상태를 정확하게 파악할 수 있으며 연속성을 갖는 웹뷰의 유지, 보수를 할 수 있다. 그리고 웹의 운영환경이 동적으로 변하는 상태에서 웹 웨어하우스 (web warehouse)에서의 웹뷰를 효율적으로 지원할 수 있다고 하고 있다. 그러나 이 연구의 경우 실제 응용에 연계되어 있는 웹 (connected application web) 환경 [8]을 기반으로 하며, 제시된 MEDI 방식과 VMF 방식에서의 각 상태에 대한 정확한 정의 없이 각 상태를 분류하고 있고, 웹뷰의 연속성 (continuity)를 기반으로 하여 생성과 폐기라는 웹뷰 콘텐츠 관리에 대한 기본적인 개념이 없어 지역적으로 분산된 환경에서의 웹뷰의 사용 상황을 파악할 수 없어 이에 대한 확장과 더불어 이에 대한 평가를 필요로 한다.

2. 상태 전이 그래프

본 연구에서의 주 관심대상인 상태 전이 그래프는 웹뷰의 상태를 파악하기 위하여 웹 페이지에 사용된 웹뷰의 상태를 항상 새로운 데이터로 보유한다. 기반 데이터에 대한 어떠한 갱신도 웹뷰에 대하여 즉각적으로 활성화 (refresh)시키게 되는 활동 상태 (A; active state)와 웹뷰가 어떠한 상태라도 갱신에 대하여 응답하지 않으며, 요청 시에만 평가를 하게 되는 휴면 상태 (S; sleeping state), 그리고 활동 상태와 휴면 상태의 중간에 위치하며 최신의 웹뷰 상태를 유지하면서 사용자에 의하여 직접 사용이 가능한 중앙값 (M; Median)의 세 가지의 상태로 파악하여 보여주는 MEDI (MEDIan) 방식과 MEDI 방식의 웹뷰의 상태에 반-활동 (SA; semi-active) 그리고 반-휴면 (SS; semi-sleeping)의 상태를 추가하여 다섯 가지 상태로 파악하여 보여주며 연속성의 법칙 (continuity law)을 가장 잘 이용하는 VMF (view maintenance based on a five-state graph) 방식이 있다. 여기에서의 반-활동과 반-휴면은 기본적으로 연구 자체에서 정확하게 정의되고 있지 않으나, 그림 1.에서의 표현으로 볼 때 활동 상태와 휴면 상태에 대한 중간값 즉, 연속적인 갱신 또는 질의에 대한 버퍼 값을 저장하고 이에 대한 운영을 지원하는 것으로 본다.

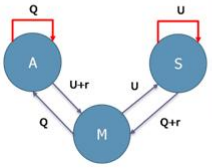

그림 1. MEDI 방식
Fig. 1. MEDI Approach

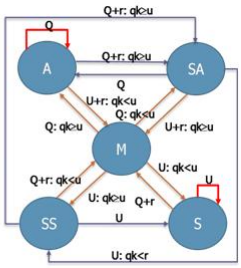

그림 2. VMF 방식
Fig.2. VMF Approach

VMF 방식은 MEDI 방식에 대한 성능 향상 방식으로 연속성의 법칙을 이용하여 높은 시스템 효율성을 보이고 있으나, 실제 분산 또는 새로이 적용되는 웹 환경과 웹뷰의 생성과 폐기에 대하여는 언급하고 있지 않아 단일 운영 환경에서의 웹뷰 상태를 제외하고는 정확한 상태를 보여주기는 힘들다. 또한 변경이 빠르게 일어날 경우는 MEDI 방식이 좋은 결과를 얻는 것으로 웹이 지속적으로 변경되며 질의 역시 급속하게 변경된다면 VMF 방식을 사용하라고 하였으나, 두 가지 모두 단일 환경에서 약간의 동적인 환경을 제공하는 상태로 볼 수 있다.

III. 상태 전이 그래프의 확장

일반적으로 새로운 웹 페이지는 주당 8% 정도의 비율로 생성되며, 갱신은 약 15% 정도만 지속적으로 일어나며, 50% 정도는 거의 변경되지 않으므로 [9] 지금과 같은 환경에서 어떤 방법이 효율적이라고는 말할 수 없다.

현재 대다수의 연구에서는 실행 중인 웹뷰 환경에서의 상태 전이를 하기 위한 조건을 제시하지 않았다. 이는 웹뷰가 웹 페이지를 구성하는 일부이며, 구성되는 일부가 지속적으로 사용되거나 갱신된다는 관점 외에 웹뷰의 생성과 이를 폐기시켜야 하는 상태 (state)에 대하여는 정의하고 있지 않다. 또한 사용한 웹뷰라고 하더라도 저장된 웹뷰의 콘텐츠를 무한정 시스템에서 보유하는 것은 동적인 웹 환경에서 문제가 있다. 즉 웹뷰 콘텐츠를 가지고 있는 웹 페이지에서의 웹뷰 콘텐츠의 쓰고 버리기 (service and waste) 그리고 쓰고 저장하기 (service and store) 개념과 더불어 생성된 웹뷰 콘텐츠의 미사용으로 인한 콘텐츠의 제거 방법에 대하여도 언급하지 않았다. 덧붙여서 이전의 연구들이 다양한 기반 데이터로부터 형성된 다양한 웹뷰 콘텐츠를 병합하여 새로운 형태의 웹뷰로 생성하기 위해 분산되어 있다는 관점이 배제된 한정된 영역에서의 상태 전이만을 언급하여 따로 조건을 제시할 필요가 없고 실제 웹뷰의 생성과 폐기 알고리즘은 다음의 알고리즘 1과 2와 같이 표현한다.

```
Proc. Create WebView
  Create_View(Base_Data,View);
  Create_WebView(View,WebView);
  Do until User_Request
     Store_Temp(WebViewn(Content));
     else
       State_Transfer(State_Requester);
       send WebViewn(Content);
       end
   end
```

알고리즘 1. 웹뷰 생성
Algorithm 1. Create of WebView

```
Proc. Obsolete_WebView
  delete (WebViewn(Content));
  end
```

알고리즘 2. 웹뷰 삭제
Algorithm 2. Obsolete of WebView

1. 역할

상태 전이의 조건은 기반 데이터를 이용, 생성된 웹뷰 콘텐츠가 해당 웹 페이지에서 사용여부를 확인하여 조건을 결정하게 된다. 이에 해당하는 상태로는 그림 3.에서와 같이 웹뷰를 사용하고자 하는 상태 요구 (state requester), 요청된 웹뷰의 콘텐츠를 제공할 수 있는 준비 단계인 상태 준비 (state provider), 그리고 생성된 웹뷰 콘텐츠는 존재하나 여러 가지 조건으로 인하여 웹 페이지의 구성에는 참여하지 않는 비 참여 (non-participant)의 세 가지 역할 (role)로 존재하며 역할 자체가 웹뷰의 사용 및 적용 상태를 설명하고 있다.

상태 요구와 비 참여는 새로운 웹뷰 콘텐츠와 웹뷰 콘텐츠 생성에 대한 상태가 생성되며 이중 상태 요구는 생성되어 저장되어 있는 웹뷰 콘텐츠를 기반으로 사용자의 요구를 확인하여 저장되어 있는 웹뷰 콘텐츠를 웹 페이지에서 사용할 수 있도록 전송하고 이에 대한 상태 제시를 하여 상태 준비 단계로 상태가 전이한다. 전이된 상태는 웹 페이지의 구성을 제시하고 이 상태에서 새로이 생성된 웹뷰는 임시 저장 장치에 저장되어 추후의 응용 즉 웹 페이지의 구성에 사용된다. 비 참여는 일정 시간 대기하다 요청되는 상태 준비로 전이되지 않으면 자동 삭제된다.

```
Proc. State_Requester
  Do until WebViewn(Content)=True
    Request_State(State_Ready)
    send WebViewn(Content))
    State_Transfer(State_Ready)
    else
      Request_State(Non-Participation)
      end
    end
```

알고리즘 3. 웹뷰 상태 요청
Algorithm 3. Request the WebView Status

```
Proc. State_Ready
  Do until Web_Page_Request = True & n_Time then
    send WebViewn(Content))
    State_Transfer(Create_Web_Page)
    else Web_Page_Request = False & On_Time then
    goto Main
    else
      Obsolete
      end
    end
  end
```

알고리즘 4. 웹뷰 상태 준비
Algorithm 4. Ready State

```
Proc. Non-Participant
  Receive_Condition(User_Request)
  Do until Check_Condition(WebViewn(Content))=
                                True & On_Time
    Request_State(State_Ready)
    send WebViewn(Content))
    State_Transfer(State_Ready)
    else if check_condition=Fales & In_Time
      Reserve_State(WebViewn)
      else out_time then
        Obsolete(WebViewn(Content))
        end
      end
    end
```

알고리즘 5. 웹뷰 상태 비 참여
Algorithm 5. Non-Participation

```
Proc. Create_Web_Page
  Receive(WebViewn(Content))
  Create_Web_PageΣ WebViewn(Content))
  Store
  end
```

알고리즘 6. 웹 페이지 생성
Algorithm 6. Creation of Web Page

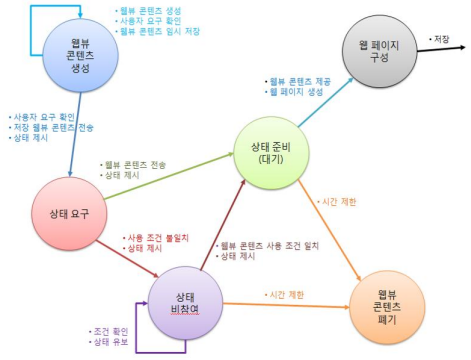

그림 3. 확장된 상태 전이 그래프에서의 역할
Fig. 3. Role in the extended State Transfer Graph

2. 공유

뷰에 있어 공유 (sharing)는 웹 페이지의 사용 목적에 따라 사용되는 문제로 생성되어 상태 준비에 상태를 제시한 웹뷰는 자신들의 목적에 따라 콘텐츠를 공유하여 새로운 콘텐츠를 생성을 위한 조건을 제시한다. 즉, 콘텐츠의 사용 정도에 따른 정도 (degree)의 설정으로 강한 (degree of strong) 공유와 약한 (degree of weak) 공유로 규정하는데 강한 공유는 콘텐츠의 갱신에 대한 엄격한 규정을 통하여 갱신과 재-계산을 최소화시켜 갱신을 제한하며, 약한 공유는 갱신의 조건을 풀어서 콘텐츠 사용 중의 갱신 또는 재-계산만 아니면 언제나 갱신과 재-계산을 허용한다. 또한 강한 공유와 약한 공유 속성을 가지는 두 가지 이상의 콘텐츠를 공유할 경우는 사용자 또는 시스템의 설정에 따라 부분적으로 갱신과 재-계산을 허용한다.

이를 그림으로 표시하면 그림 4.와 같으며 먼저 웹뷰 콘텐츠가 강한공유와 강한공유의 속성을 가지고 있으면 갱신을 할 수 없으며, 강한 공유와 약한 공유의 속성을 가지고 있을 경우는 제한적인 갱신만을 허용하며, 약한 공유와 약한 공유 속성을 가진 웹뷰 콘텐츠들은 기본적으로 모든 갱신을 허용하게 되나, 이때 잘못된 공유로 인한 오류 발생을 방지하기 위하여 웹 운용 환경을 사용자 또는 시스템의 설정을 통하여 결정하게 된다

.

```
Proc Sharing
  If Degree = Strong(WebViewn(Content))
    Degree = Strong(WebViewn(Content))then
    limited_update(WebViewn(Content)
    limited_re-compute(WebViewn(Content)
     else if Degree=Strong(WebViewn(Content)
      Degree=Weak(WebViewn(Content))then
       if Check_Condition (system)= true
         limited_update (WebViewn(Content)
         limited_re-compute(WebViewn(Content)
        else
         stop
       end
       else
         update(WebViewn(Content))
         end
      end
```

알고리즘 7. 웹뷰에서의 공유
Algorithm 7. Sharing in the WebView

3. 시간

시간 (time)은 웹뷰의 생성과 폐기에 있어 매우 중요한 조건이다. 특히 웹뷰의 생성 조건이 분산 환경이라면 시간 설정에 따라 웹뷰 콘텐츠의 생성과 폐기 또는 공유의 강도를 재설정하여야 한다. 더구나 외부적인 요건에 의한 지연으로 인하여 콘텐츠 간의 조인 작업에 문제가 발생한다면 콘텐츠의 재-계산이 불가능하다. 어느 정도의 설정에 따라 지연에 의한 문제가 해결되지 않는다면 웹뷰의 생성 자체를 폐기하고 제시된 상태의 회수 작업을 수행한다.

- 질의 지연

기반 데이터의 검색을 위한 것으로 다양한 기반 데이터를 수집하여 사용자가 원하는 콘텐츠로 재구성하기 위하여 필요하며, 재-계산은 새로운 웹 환경에서는 지속적으로 발생하게 된다. 질의 지연의 개선은 사용자에 의한 기반 데이터 검색의 횟수의 제한을 통하여 가능하나 이 경우 적정한 검색 횟수의 설정 방식에 문제점이 발생할 수 있어 신중한 처리가 필요하다.

- 연속적인 요청에 의한 재-계산

웹뷰 콘텐츠의 갱신이 완료된 시점에 사용자에게 콘텐츠의 사용을 허락하여 지연 문제가 발생하게 되는데, 이를 통하여 웹뷰 콘텐츠를 이용한 웹 페이지의 구성에 오류가 발생하지 않도록 한다. 물론 웹 페이지의 구성에서 발생할 수 있는 오류 중 집단 지성 (Group Thinking)에 의하여 발생하는 오류에 의한 부분은 여기에 해당하지 않는다.

- 갱신 중 오류 발생으로 인한 전송 지연

웹뷰의 갱신 중 발생하는 오류는 기존의 웹뷰를 폐기하고 새로운 웹뷰 콘텐츠의 읽어 오기 및 생성을 요구하는 과정으로 인한 지연으로 웹뷰의 정확도 확인을 어디까지 설정할 것인가가 문제가 된다.

- 갱신된 웹뷰의 콘텐츠를 사용 중 갱신으로 인한 콘텐츠 비교 발생 지연

이러한 경우는 사용자 또는 시스템의 설정에 의하여 사용 중인 웹뷰에 대한 사용 중 갱신을 방지할 수 있으나, 실시간으로 웹 페이지의 웹뷰 콘텐츠가 변경되는 경우는 사용 중인 콘텐츠의 변경으로 인한 재계산이 불가피하다. 실시간으로 웹 페이지의 웹뷰 콘텐츠가 변경되는 경우는 시간의 차를 두고 웹뷰 콘텐츠의 갱신을 시도하여 불필요한 콘텐츠 비교 지연을 방지하여야 한다.

- 분산된 환경에서의 네트워크 지연으로 인한 재-계산

지연하드웨어적인 문제 또는 외부적으로 발생하는 것으로 내부 처리가 중요한 것이 아니라 외부적인 조건에 대한 가중치 처리 문제가 중요하다. 가중치는 기본적으로 웹 페이지에서 사용되는 웹뷰 콘텐츠를 네트워크를 사용하여 가져오는 지 아니면 가지고 오더라도 사용하게 되는지 등에 대한 확인에 따라 적용이 가능한지가 확인된다.

4. 활성 상태 / 수동 상태 / 휴면 상태

웹뷰의 콘텐츠가 생성되어 웹 페이지에 사용되는 상태로 [9]의 연구에서 활성 상태 (active state)와 휴면 상태 (sleep state)로만 구분하였으나, 활성 상태는 다시 활성 상태와 수동 상태 (passive state)로 분류된다. 활성 상태는 사용자의 질의에 의하여 웹뷰 콘텐츠가 정상적인 상태에서 생성되고 상태가 상태 준비로 제출되는 상태로 콘텐츠에 대한 갱신과 재-계산이 가능하다. 활성 상태에서 사용자 질의 요청에 의한 웹뷰 재-계산 요청이 발생할 경우에는 중간값을 이용하여 질의 요청에 대한 작업을 시행하게 된다. 수동 상태는 연속 갱신 또는 기타 지연으로 인한 웹뷰 콘텐츠가 미 성숙된 상태에서 갱신과 재-계산 등의 일상적인 상태 전이를 중지하고 일정 상태에 도달한 이후 사용자에 의한 갱신과 콘텐츠의 재-계산을 요청하며, 활성 상태인 웹뷰 콘텐츠에서도 시스템의 불안 등의 내부 원인으로 인한 경우는 수동 상태로 전이시켜 이에 대한 대비를 하도록 한다. 휴면 상태는 일정 시간 동안 데이터의 급격한 변화가 없는 상태로 일정 시간 이상 지속적으로 수동 상태로 존재할 경우는 시스템의 효율성을 따져 폐기 여부를 고려한다.

```
Proc Create_Web_Page
  if WebView(State)=Active then
    if Complete(Create_Web_Page) = True then
    Create_Complete_Web_Page(WebView_n(Content))
    Request_User_Query
    Update(Web_Page(WebView_n(Content))
    Re-Write(Web_Page(WebView_n(Content))
    MIDi(Web_Page(WebView_n(Content)))
    else
      State_Transfer(Passive_Stat)
      end
    end
    elseif WebView(State)=Passive then
      if time_limit = true then
        if Complete(Create_Web_Page) = True then
          Trasfer(WebView_n(Content))
          State_Transfer(Active_Statet)
          end
          else
           Update(Web_Page(WebView_n(Content))
           ReWrite(Web_Page(WebView_n(Content))
           end
        end
        else verify(WebView_n(Content))
          Obsolete_WebView(WebView_n(Content))
          end
        end
      else Request_User_Update
        Update(Web_Page(WebView_n(Content))
        MIDi(Web_Page(WebView_n(Content)))
      end
    end
```

알고리즘 8. 활성, 수동 및 휴면 상태
Algorithm 8. Active, Passvie and Sleep state

```
sub MIDi(Web_Page(WebView_n(Content))
  if MIDi(State)=Active then
    request_Query(Web_Page(WebView_n(Content))
    else request_Update(Web_Page(WebView_n(Content))
    end
  end
```

알고리즘 9. 중간값의 저장과 갱신
Algorithm 9. Store and update of Meidan

5. 크기

크기 (size)는 웹 페이지에서 사용된 웹뷰의 복잡도에 따라 상이하다. 정적이거나 동일한 운영환경에서의 웹뷰는 복잡도가 크지 않으나, 분산 환경의 경우 지역적으로 분산된 데이터베이스에서 기반 데이터를 읽어 사용자의 질의에 합당하는 웹뷰 콘텐츠를 생성하므로 시간적인 제약 문제가 발생하게 된다. 그리고 입력되어 있는 데이터의 양과 더불어 읽어온 데이터의 비교 분석을 통한 병합 (join) 작업의 경우 작업이 이루어지는 하드웨어의 성능에 따라 사용자가 요구하는 웹 페이지 생성의 효율성을 좌우하게 되어 성능에 영향을 미치는 요인들을 확인하여 미리 제시하여야 하며, 병합의 결과를 판단하고 이에 대한 의미와 지식으로의 사용은 기본적으로 시맨틱 웹과 메타웹 환경에서의 웹뷰 콘텐츠 활용으로 본다.

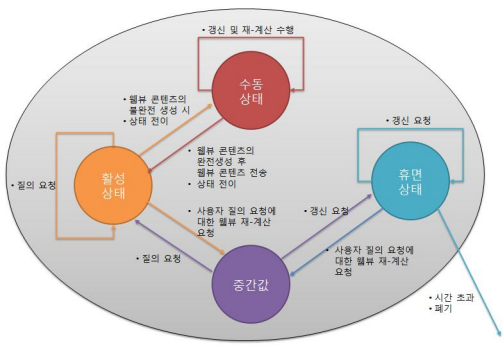

그림 5. 활성 상태 / 수동 상태 / 휴면 상태의 확인
Fig. 5. Confirmation of active, passive and sleep state

6. 상태 방해 / 비방해

정해진 시간 내에 웹뷰 콘텐츠 생성 작업 또는 갱신과 재-계산 작업이 완료되지 않아 수동 상태로 진입하였을 경우는 상태를 상태 방해 (blocking)로 설정하고 작업이 완료될 때까지 모든 갱신 또는 재-계산 작업을 중지하고 그림 6에서와 같이 수동 상태를 완료하고 활성 상태로 전이한다. 그러나 사용자 또는 시스템의 설정 및 판단에 의하여 웹뷰 콘텐츠의 생성이 완료되지 않은 상태에서도 갱신과 재-계산을 수행되면 상태 비방해 (non-blocking)로 설정하여 상태의 전이를 제시하고 갱신과 재-계산을 사용자와 시스템의 설정에 따라 진행한다. 이때 수행의 결과는 사용자와 시스템의 설정 오류로 의하여 부정확할 수 있으므로 이에 대한 검증이 필요로 한다.

```
Proc. Blocking_n_Non-Blocking
  if WebView(State)=Passive &
       Complete(Create_Web_Page)=False then
   if User_Query(Update(Web_Page(WebViewn(Content))
      or
     User_Quer(Rewrite(Web_Page(WebViewn(Content))
     then
       State_Transfer(Passive,non-Blocking)
     else
       State_Transfer(Passive,Blockingt)
       Do until Complete(Create_Web_Page) = True
      end
    end
```

알고리즘 10. 웹뷰 생성의 방해 및 비방해
Algorithm 10. Blocking and non-blocking in the WebView creation

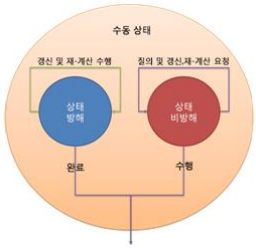

그림 6. 수동 상태에서의 상태 방해와 비방해
Fig. 6. Blocking and Non-Blocking in the passive state

7. 동기화

활성 상태에 진입한 웹뷰의 콘텐츠에 많은 사용자가 동시에 사용하거나 갱신을 요청할 경우 이에 대한 동기화 (synchronization) 상태를 확인하여야한다. 순간적으로 기존 웹뷰 콘텐츠의 변경이 발생한 상태이면 이전의 웹뷰 콘텐츠의 상태를 버전 (version) 단위로 설정, 사용하며, 버전은 갱신 또는 재-계산된 웹뷰와 더불어 저장된 (stored) 웹뷰 또는 복제 (replicated)된 웹뷰에도 동일하게 버전을 부여하게 된다. 버전의 관리는 상태 전이에서 시행하게 되며, 각 버전에는 특정의 번호를 부여하게 된다. 특히 웹뷰의 각 버전에 설정된 현재 상태에 따라 그림 7.과 같은 새로운 버전의 생성과 지연 가능성을 예측할 수 있다.

```
Proc. Version
  if WebView(State)=Active
   if User_Query(Update) then
     Update(WebViewn(Content)=WebViewn+1(Content)
     elseif User_Query(Backup)then
       Backup((WebViewn(Content)=
                          WebViewn+1(Content)
     Restore(WebViewn+1(Content))
     elseif User_Query(Replicate)then
      Replicate((WebViewn(Content)=
                          WebViewn+1(Content)
      else
       do nothing
       end
     end
   end
  else
   do nothing
   end
  end
```

알고리즘 11. 웹뷰의 버전 설정
Algorithm 11. Set the versions of WebView

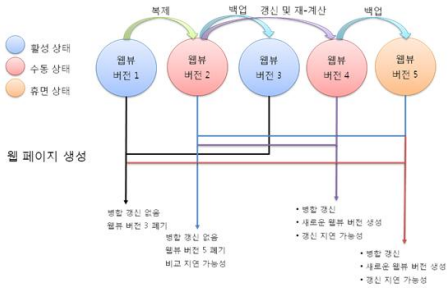

그림 7. 버전의 상태에 따른 웹뷰 상태의 변화
Fig. 7. WebView State changes which follows in Version Status

그림. 7에서의 버전의 상태에 따른 웹뷰 상태의 변화를 정리하면 먼저 새로이 사용하게 되는 서로 상이한 웹뷰 콘텐츠 간의 결합 생성, 복제된 웹뷰 버전들간의 결합 생성, 활성 상태의 웹뷰 콘텐츠와 또 다른 활성 상태의 웹뷰 콘텐츠간의 결합 생성, 활성 상태의 웹뷰 콘텐츠와 수동 상태의 웹뷰 콘텐츠 간의 결합 생성 등 다양한 종류의 결합된 웹뷰 콘텐츠를 생성한다. 이때 생성된 웹뷰의 상태에 따라 폐기가 되는 웹뷰 콘텐츠, 웹뷰 콘텐츠의 병합 생성 발생 또는 병합 갱신의 발생/미발생, 새로운 웹뷰의 생성과 더불어 갱신 지연, 비교 지연 등의 발생 여부를 확인할 수 있다.

8. 상태의 백업; 복제와 스냅샷

상태의 백업 (backup)은 웹뷰의 생성 또는 갱신, 재-계산 도중의 발생한 문제로 인하여 생성되던 웹뷰의 상태를 저장하여 이에 대한 추후 재시작에 대비하도록 한다. 그러나 확장된

상태 전이에서는 상태의 저장이 현재 운영 중인 상태에 대한 저장이며, 이 상태의 실제 운영 상황 및 중간값에 대한 것은 저장되지 않는다. 이를 위하여 현재 운영 상태의 복제(duplication)를 지속적으로 시행하여 추후 발생이 가능한 시스템의 운영 중지로 인한 저장되었던 중간값의 상실을 방지하며, 추후 웹뷰 콘텐츠를 이용한 새로운 웹 페이지의 구성에 대비한다. 지금까지의 상태에 대한 저장을 통하여 추후 시스템의 재개 시에 이제까지 사용하였던 상태의 저장된 스냅샷(snapshot)을 통하여 중지되기 바로 전의 상태 전이를 복구하고 복제되었던 상태의 콘텐츠를 다시 사용하여 원래의 웹 페이지의 구성작업을 수행한다. 알고리즘. 13에서는 웹뷰의 상태가 오류가 발생하였음을 알리는 플레그가 설정되면서부터 현재 상태와 내용을 저장하는 작업을 수행함을 보여주며 두 번째 알고리즘은 웹뷰의 진행이 복구된 이후 다시 이전의 저장된 복제와 스냅샷을 복구시키는 작업을 표시한다.

```
Proc. Snapshot
  if Create WebView = False then start
   Set Restore_State = False
     Dump(WebView_current(Content))
          Store
     Dump(WebView_current(State))
          Store
     end
```

알고리즘 13. 웹뷰 스냅샷
Algorithm 13. Snapshot of WebView state

8. 상태의 백업; 복제와 스냅샷

상태의 백업 (backup)은 웹뷰의 생성 또는 갱신, 재-계산 도중의 발생한 문제로 인하여 생성되던 웹뷰의 상태를 저장하여 이에 대한 추후 재시작에 대비하도록 한다. 그러나 확장된 상태 전이에서는 상태의 저장이 현재 운영 중인 상태에 대한 저장이며, 이 상태의 실제 운영 상황 및 중간값에 대한 것은 저장되지 않는다. 이를 위하여 현재 운영 상태의 복제 (duplication)를 지속적으로 시행하여 추후 발생이 가능한 시스템의 운영 중지로 인한 저장되었던 중간값의 상실을 방지하며, 추후 웹뷰 콘텐츠를 이용한 새로운 웹 페이지의 구성에 대비한다. 지금까지의 상태에 대한 저장을 통하여 추후 시스템의 재 개시에 이제까지 사용하였던 상태의 저장된 스냅샷 (snapshot)을 통하여 중지되기 바로 전의 상태 전이를 복구하고 복제되었던 상태의 콘텐츠를 다시 사용하여 원래의 웹 페이지의 구성작업을 수행한다. 알고리즘. 13에서는 웹뷰의 상태가 오류가 발생하였음을 알리는 플레그가 설정되면서부터 현재 상태와 내용을 저장하는 작업을 수행함을 보여주며 두 번째 알고리즘은 웹뷰의 진행이 복구된 이후 다시 이전의 저장된 복제와 스냅샷을 복구시키는 작업을 표시한다.

```
Proc. Restore_State
  if Restore_State = True then start
    Restore(WebView_before(State))
    Restore (WebView_before(Content))
    Set Create WebView = True
    else
     Restore_State = False
     loop
     end
    end
```

알고리즘 14. 웹뷰 스냅샷 복구
Algorithm 12. Recovery of WebView snapshot

IV. 알고리즘에 대한 검증

제안된 알고리즘은 사용자들이 필요로 하는 기반 데이터들이 지역적으로 분산되어 있는 데이터베이스 사이트에 존재하고 있는 상태에서 사용자들에 의한 질의를 통하여 웹뷰 콘텐츠가 생성하여 웹 페이지에서 사용하는 과정을 설명하고 있다. 그러나 이 과정에서 확인해볼 필요가 있는 부분이 지역적으로 분산되어 있는 환경에서 구체화 웹뷰를 사용하였을 경우와 사용하지 않았을 경우 그리고 네트워크의 상태에 따른 시간 분석과 데이터의 크기에 따른 효율성에 대한 분석으로 이를 위하여 다음과 같은 환경에서 실험을 실시하였다.

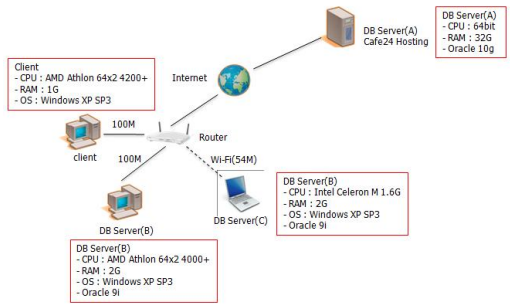

그림 8. 시스템 구성
Fig. 8. System Configuration

본 실험의 기본적인 장비는 지역 네트워크 환경에서 2대의 컴퓨터와 외부 데이터베이스 호스팅 서버에 오라클 데이터베이스 서버를 구축하였다. 해당 데이터베이스를 사용하는 각 클라이언트와 데이터베이스 서버 B는 100M 속도를 지원하는 이더넷 네트워크로 데이터베이스 서버 C는 무선 네트워크로

연결하였다. 그리고 각 데이터베이스 서버에 동일한 개수의 데이터를 입력한 후 구체화 웹뷰를 적용하였을 경우와 웹뷰를 적용하지 않았을 경우의 결과를 읽어오는 시간을 비교하였다.

먼저 지역 네트워크 환경의 데이터베이스 서버에 대한 구체화 웹뷰의 사용 및 미사용에 따른 처리 시간의 변화를 보게 되면 다음의 그림. 9와 그림. 10에서와 같이 구체화 웹뷰 적용과 비적용 모두 데이터의 개수가 증가할수록 처리 시간이 급격하게 증가하지만 구체화 웹뷰의 경우는 새롭게 추가된 부분만 처리해주면 되기 때문에 시간 증가의 정도가 구체화 웹뷰를 적용하지 않았을 경우와 비교했을 때 아주 미미한 정도의 처리 시간의 증가를 보이고 있으며 각 서버의 하드웨어 성능이 좋을수록 처리 시간이 줄어드는 것을 볼 수 있다.

다음으로 그림. 11과 그림. 12에서는 외부 서버를 이용한 구체화 웹뷰의 사용 및 미사용에 따른 처리 시간의 변화를 보여주고 있는데, 여기에서 먼저 지역적으로 분산되어 있는 환경이기 때문에 사용자의 질의에 대한 웹뷰 콘텐츠를 생성하기 위하여는 먼저 서버에서 웹뷰의 생성을 실시하게 되며, 이 경우 하드웨어의 성능이 좋을수록 구체화 웹뷰를 적용하거나 적용하지 않거나 거의 사용자가 요구하는 콘텐츠를 생성하는데 유사한 내부 처리 속도를 제공하게 되나 지역적으로 분할되어 있다는 특성으로 인하여 실제 사용자가 느끼는 수행 시간은 상대적으로 오래 걸리는 것으로 나타났으나, 실제 구체화 웹뷰를 적용한 경우는 처리 시간이 일정하게 변하지 않으나 적용하지 않은 경우를 경우 데이터의 처리 용량이 증가함에 따라 처리 시간이 늘어나는 것을 확인할 수 있었다.

이와 같은 실험의 결과에서 구체화 웹뷰의 사용과 미사용에 따라 생성된 웹뷰 콘텐츠의 생성된 데이터 크기와 네트워크의 사용 여부 그리고 웹뷰 콘텐츠를 관리하는 장비의 특성에 의하여 콘텐츠를 사용자 웹 페이지에서 효율적으로 사용할 수 있는지의 여부를 결정할 수 있음을 확인할 수 있었으며, 위의 실험에서 웹뷰 콘텐츠의 개수가 많지 않을 경우는 네트워크의 사용, 미사용에 상관없이 구체화 웹뷰를 사용하지 않더라도 처리 시간에 있어 큰 차이가 발생하지 않았으나, 사용하게 되는 웹뷰 콘텐츠의 크기가 증가함에 따라 처리 시간도 증가함을 볼 수 있어 데이터의 개수가 많지 않을 경우는 웹뷰를 사용할 필요가 없음을 알 수 있었으며, 네트워크의 사용 여부는 일단 구체화 웹뷰 콘텐츠의 갱신이 각자 자체적으로 이루어진 상태에서 결과 데이터만을 가지고 오게 되어 네트워크가 단절되어 있는 상황이 아니라면 실험에서는 큰 의미를 부여할 수 없다.

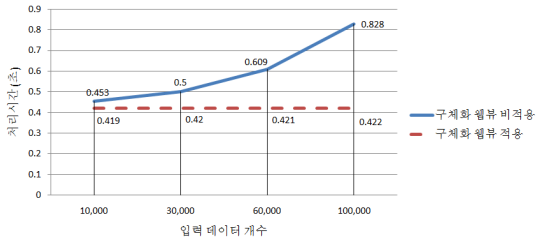

그림 9. 데이터베이스 서버 B에서의 처리 시간 변화
Fig. 9. Variation of process timing in DB server B

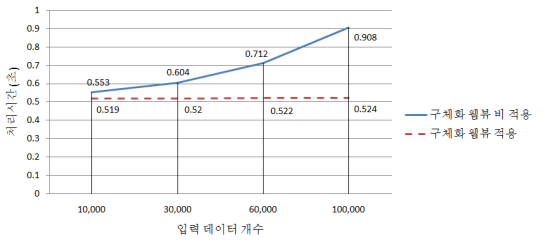

그림. 10 데이터베이스 서버 C에서의 처리 시간 변화
Fig. 10. Variation of process timing in DB server C

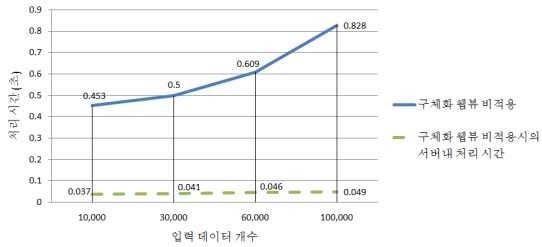

그림. 11. 구체화 웹뷰 사용하지 않은 데이터베이스 서버 A에서의 내부 및 외부 처리 시간 변화
Fig. 11. Variation of processing time in DB server A (do not use the materialized WebView)

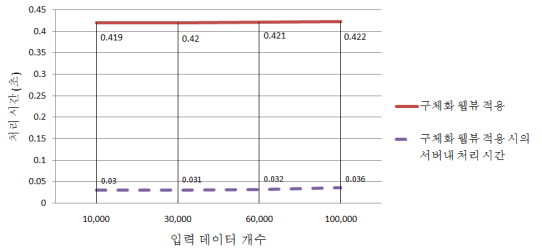

그림. 12. 구체화 웹뷰 사용한 데이터베이스 서버 A에서의 내부 및 외부 처리 시간 변화
Fig. 12. Variation of processing time in DB server A (use the materialized WebView)

V. 결 론

본 연구에서는 [5]의 연구에서 제시되었던 상태 전이 그래프의 확장을 통하여 새로운 웹 환경에서의 웹뷰 모델의 적용을 위한 확장된 상태 전이 그래프 모델과 이를 수행하기 위한 알고리즘을 제시하여 새로운 웹 환경에서의 웹뷰 상태를 확인할 수 있도록 하였다. 또한 제안된 알고리즘을 적용한 웹뷰 콘텐츠의 갱신과 사용에 대한 예를 통하여 실제 웹뷰가 사용되는 환경에서의 웹뷰 상태 확인 방법에 대하여 확인하였다. 실제 확장된 상태 전이 그래프를 기존의 연구에 적용할 경우 대부분의 웹뷰 상태를 설명할 수 있으며, 이를 통한 웹뷰 구체화 모델의 비용의 계산을 손쉽게 처리할 수 있을 것으로 기대하며, 실제 제안된 알고리즘 중 시간, 네트워크 그리고 웹뷰 콘텐츠의 크기에 대한 실험에서 웹뷰 콘텐츠의 크기에 따라 처리 시간에 많은 차이를 발생시킬 수 있으나, 크기가 적을 경우는 큰 차이를 보이지 않아 구체화 웹뷰를 적용할 대상의 선택에 신중을 기하여야 한다는 것을 확인할 수 있었다. 또한 지역적으로 분산되어 있음이 네트워크를 사용하고 있다는 것을 의미하는데 이에 따라 구체화 웹뷰의 적용에는 큰 문제점을 보이지 않았고 전송 속도에 있어서만 차이가 있음을 확인할 수 있어 네트워크 전송 통로의 선택이 매우 큰 영향을 미칠 수 있음도 확인할 수 있었다. 또한 원격지 사이트에서의 하드웨어 성능이 전체 웹뷰의 관리에 많은 영향을 끼칠 수 있음도 실험을 통하여 확인할 수 있었다.

추후 본 연구의 알고리즘 중 웹뷰 콘텐츠의 병합에 의한 사용자 웹뷰의 생성에서는 지역적으로 분산된 상태에서 어느 위치에서 웹뷰 콘텐츠의 병합을 실시할 것인지를 결정하여야 하며, 결정되는 위치에 따라 웹뷰 콘텐츠의 생성 지연 또는 기타 오류의 발생에 영향을 미칠 수 있기 때문으로 이에 대한 추가 연구가 필요하다.

참고문헌

[1] G.Huston, "Web Caching," Cisco. The Internet Protocol Journal - Volume 2, No. 3. Sept. 2009

[2] A.Vakali, "Evolutionary Techniques for Web Caching," Distributed and Parallel Databases, Vol. 11, No. 1, pp73-92 Jan. 2002

[3] A.Labrinidis and N.Roussopoulos, "On the Materialization of WebViews," In Proc. of the ACM SIGMOD Workshop on the Web and Databases (WebDB'99), 1999

[4] J.Forsythe, X.Ke, and L.Oats, "A Four-Tier Model of A Web-Based Book-Buying System," May 2004.

[5] Y.Zhang and X.Qin, "State Transfer Graph: An Efficient Tool for WebView Maintenance," In the Proc. of WAIM2005, Hangzhou, China. October 2005

[6] A. Labrinidis, "View Materialization for The Web," Dissertations for Ph.D, University of Maryland, 2002

[7] 서진형, "분산 네트워크 환경에서의 웹뷰 구체화," 한국정보과학회 29차 추계학술대회, 2002년 10월

[8] 서진형, 김경창, "분산환경에서의 WebView 구체화 모델을 위한 사용자 서비스 확장," 제 19 회 한국정보처리학회 춘계학술발표대회 춘계학술발표, 2003년 9월

[9] A.Ben Ammarj, A.Abdellatif, and H.B.Ghezala, "Forms of Data Materialization in Data-Intensive Web Sites," IJCSNS, VOL.7 84 No.12, pp84-88, December 2007

[10] J.H.Suh and W.H.Lee, "A New Model for Materialized WebView in Web Environment," 2009 ICMIT, December 3-5, 2009

[11] A. Ntoulas, J.Cho, and C.Olston, "What's new on the web? the evolution of the web from a search engine perspective," In Proc. of the WWW, May, 2004.

저 자 소 개

서 진 형

경인여자대학 정보미디어과 교수

1986년 2월 : 홍익대학교 전자계산학과 졸업 (학사)

1998년 8월 : 홍익대학교 대학원 전자계산학과 졸업 (석사)

2010년 2월 : 한세대학교 대학원 유시티 IT 산업정책학과 IT 전공 졸업 예정 (박사)

한국 HP, SGI 시스템 엔지니어

관심분야 : 웹, 컴퓨터그래픽스, 멀티미디어, 데이터베이스 등

이 왕 헌

한세대학교 IT 학부 교수

1985년 2월 : 서울대학교 제어 계측공학과 졸업(학사)

1995년 2월 : KAIST 자동화 및 설계 공학과 졸업(석사)

2001년 8월 : KAIST 자동화 및 설계 공학과 졸업(박사)

관심분야 : 임베디드 컨트롤러, 초음파 센서기반의 로봇 제어, 비전기반의 이동로봇 주행제어, 비쥬얼 서보잉등