컴포넌트 기반 미들웨어 자기최적화와 자가치료을 위한 베이지안 네트워크를 사용한 시스템 자원 상태 추론

최보윤[O] 김경중 조성배
연세대학교 컴퓨터과학과
bychoi@sclab.yonsei.ac.kr[O], kjkim@cs.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

Inference of System Resource States Using Bayesian Network for Self-Optimizing and Self-Healing Component-based Middleware

Bo-Yoon Choi[O], Kyung-Joong Kim, Sung-Bae Cho
Dept. of Computer Science, Yonsei University

요 약

최근 컴포넌트 기반 미들웨어의 최적화에 대한 연구가 활발히 이루어지고 있다. CPU점유율이 높은 어플리케이션의 동시 실행은 시스템에 부하를 주기 때문에, 시스템 성능을 약화시키고 실행중인 어플리케이션에 영향을 준다. 컴포넌트 기반 미들웨어는 여러 개의 재사용 가능한 컴포넌트를 조합하여 어플리케이션을 구성하기 때문에 동적으로 재구성이 가능하다. 본 논문은 컴포넌트 기반 미들웨어가 시스템 상황에 대한 정보를 받아들여 시스템의 상황을 스스로 판단하고 자가치료 또는 시스템의 성능을 최적화시키는 컴포넌트를 선택하는 방법을 제안한다. 상황판단을 위해 유연한 추론이 가능하고, 데이터로부터 자동학습이 가능한 베이지안 네트워크를 사용하였다. 두 시간 가량의 데이터를 리눅스 사용자로부터 획득하여 실험한 결과, 테스트 데이터에 대해 76.5%의 성능을 보였다.

1. 서 론

컴퓨터가 복잡해지고 그 기능이 다양화되면서 좀 더 효과적으로 컴퓨터를 사용하기 위해 멀티 태스킹이나 CPU스케줄링 등 많은 방법이 강구되어 왔다. 그러나 이전보다 복잡해진 시스템과 cpu점유율이 높은 프로세스의 증가는 스케줄링이나 멀티태스킹 기법만으로는 시스템의 불안전성 문제를 해결하기 어렵게 되었다.

기존의 방법은 시스템 부하를 유발할 수 있는 환경적 조건을 예방하는 수준의 문제해결 방법이었다. 그렇기 때문에 돌발적이고 실시간으로 일어날 수 있는 문제에 대해서는 시스템 스스로 대처할 수 있는 방법이 존재하지 않는다. 시스템은 인터넷의 발달과 그 기능의 다양화로 예전보다 빠르고 민감하게 변화에 반응한다. 시스템이 갑작스런 변화가 발생했을 때 스스로 돌발상황에 동적으로 대처하면 좀더 안정적으로 상태를 유지할 수 있을 것이다. 또한 시스템이 문제가 발생 할 수 있는 상황을 예상해서 원인을 미리 해결한다면 문제 발생율이 줄어들 것이다.

최근 컴포넌트 기반 미들웨어가 환경의 변화에 유연하게 재구성이 가능해서 많이 이용되고 있다. 베이지안 네트워크는 유연한 추론이 가능한 추론 기법이기 때문에, 동적 재구성이 가능한 미들웨어를 자기최적화 및 자가치료하는데 사용할 수 있다. 본 논문은 분산환경의 시스템에서 급작스러운 환경의 변화로 프로세스가 원활하게 동작할 수 없을 때, 컴포넌트 기반 미들웨어가 스스로 문제를 예상하고 그 문제를 해결하는 방법을 제안하고, 실험을 통하여 제안하는 방법의 유용성을 검증한다.

2. 관련연구

2.1 재구성 가능한 컴포넌트 구조

미들웨어의 구조는 모니터 단계, 평가 단계, 실행 단계의 논리적 기능으로 이루어졌다. 모니터 단계는 실행 환경과 소프트웨어 어플리케이션의 실시간 정보를 얻는 단계이다. 또한 수집된 정보와 잠정적으로 환경에 영향을 받아 문제를 일으킬 수 있는 컴포넌트를 분석한다. 평가 단계는 현재 시스템 환경과 프로세스의 상태에 대해 수집한 정보와 컴포넌트에 대한 정보를 사용하여 최적의 컴포넌트 구성을 결정한다. 이때 동일한 기능을 수행하는 다양한 컴포넌트를 미리 만들어 현재 상황에 가장 적절한 것을 택한다. 평가단계에서는 실행단계로 활성화하거나 혹은 비활성화할 컴포넌트 변수에 대한 명령을 보낸다. 실행 단계는 소프트웨어 어플리케이션과 컴포넌트 교체 알고리즘을 기술하고 있다[1].

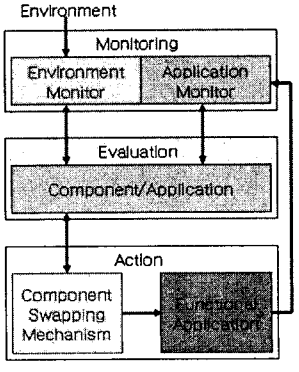

그림 1. 컴포넌트 재구성 가능 미들웨어의 구조

기존의 연구에서는 컴포넌트 교체에 대한 결정을 간단한 규칙에만 의존하고 있다[1]. 본 논문에서는 중복기반 재구성 가능한 미들웨어에서 평가단계를 베이지안 네트워크 추론 방법을 사용하여, 시스템이 스스로 판단하여 상황을 최적으로 유지하고 돌발적인 문제를 자가치료할 수 있도록 제안하였다.

2.2 베이지안 네트워크 학습

베이지안 네트워크는 방향성 비순환 그래프(directed acyclic graph, DAG)로 그래프의 각 노드는 확률변수를 나타내며, 아크는 노드들 간의 의존성을 나타낸다. 베이지안 네트워크는 불확실한 상황에서 정보를 표현하거나 추론하는 대표적인 방법으로 인공지능 분야에서 널리 사용되고 있다.

$$P(x_1, x_2, ..., x_n) = \prod_i P(x_i \mid Parents(X_i))$$

베이지안 네트워크의 구조는 추론 노드의 부모 노드 값만이 추론 노드에 조건이 된다. 베이지안 네트워크는 노드 X_1에서 X_n 까지 순서를 가진 n개의 노드를 가지고 있다. $x_1, x_2, ..., x_n$은 각 노드가 가질 수 있는 증거 값이고, X_i는 추론할 노드이다. 부모 노드는 추론 노드의 이전 노드들로 이루어진 서브집합에 포함되어있다.

$$Parents(X_i) \subseteq \{x_1, ..., x_{i-1}\}$$

베이지안 네트워크를 학습하는 문제는 주어진 평가 척도에 따라 데이터의 훈련 집합(training set) E에 가장 잘 부합되는 네트워크를 구하는 것이며, 여기서 E는 모든(또는 적어도 일부) 변수에 대한 값의 사례 집합이다. 네트워크를 구한다는 것은 DAG구조와 DAG의 각 노드에 연관된 조건부 확률 테이블(conditional probability table, CPT)을 함께 구하는 것을 의미한다. 베이지안 네트워크의 DAG구조가 알려져 있지 않은 경우에는 네트워크 구조를 평가하는 평가척도를 도입하여 네트워크 구조를 탐색해서 가장 적합한 네트워크를 구해야 한다. 하지만 모든 네트워크를 탐색하는 것은 NP-hard임이 증명되었다. 따라서 모든 네트워크를 탐색하는 것은 불가능하므로 유전자 알고리즘과 같은 휴리스틱한 방법이 많이 사용된다[2].

3. 제안하는 방법

제안하는 방법은 미들웨어 구조에서 두번째 단계인 평가단계를 베이지안 네트워크를 사용해서 최적의 컴포넌트를 추론하는 것이다. 시스템의 현재 상태와 사용중인 어플리케이션의 정보를 얻어서 몇 초 후의 시스템상황을 추론하여 적합한 컴포넌트 구성을 결정하는 방법이다. 추론은 베이지안 네트워크를 사용하였고, 추론결과를 사용하여 명령을 보내는 부분은 if-then 규칙을 사용하였다. 그림 2는 제안하는 방법의 구조이다.

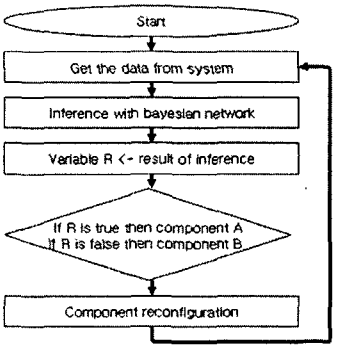

그림 2. 제안하는 방법

3.1 시스템 정보 수집 및 시스템 모니터링

실험 시스템은 일반 리눅스 컴퓨터(CPU 1.6GHz, 메모512 MB)를 사용하였다. 실험에서 베이지안 네트워크 증거 노드는 CPU정보와 실행되고 있는 프로세스 정보를 사용한다. 실험에서 시스템 정보를 일정량 수집하기 위해서 시스템 모니터링 프로그램을 실행하였다. 프로그램은 3초 간격으로 데이터를 수집하며, proc 디렉토리에서 현재 CPU정보를 보여주는 명령어를 실행한다. 또한 ps 명령어를 사용하여 실행중인 프로세스 정보를 수집하도록 구현하였다.

```
[bychoi@localhost bychoi]$ cd /proc

[bychoi@localhost proc]$ cat loadavg
0.05 0.54 0.47 1/87 20653

[bychoi@localhost proc]$ ps -u root
  PID TTY          TIME CMD
    1 ?        00:00:04 init
    2 ?        00:00:00 keventd
    3 ?        00:00:00 kapmd
```

그림 3. 현재 CPU정보와 실행중인 프로세스 정보 얻기

3.2 베이지안 네트워크 구조생성

시스템 예측을 위한 베이지안 네트워크를 학습하기 위해 시나리오를 작성하고 시나리오 대로 시스템을 사용하였다. 시나리오 재현으로 얻은 데이터는 시스템 정보(현재 CPU 상태)와 실행중 인 프로세스에 대한 정보이다. 베이지안 네트워크 구조 설계에는 전문가가 직접 설계하는 방법과 학습을 통한 방법이 있다. 시스템의 상태 추론을 위해서는 현재 시스템 상태를 파악하고, 그 정보를 이용하여 현재 시스템상황과 미래의 시스템 상황의 인과관계를 파악해야 한다. 그러나 현재 시스템 정보는 실시간으로 변하기 때문에 미래의 시스템 상황과의 보편적인 인과관계를 판단하기 어렵기 때문에 그리드 탐색과 베이지안 평가 함수를 사용하여 베이지안 네트워크 구조를 학습하였다.

조건부 확률 테이블을 생성하는 방법은 전문가가 직접 확률값을 설정하는 방법과 학습을 통한 방법이 있다. 본 논문에서는 데이터로부터 자동학습 하였다.

표 1. 시나리오

list of detection	flow of time									
	0-3	3-6	6-9	9-12	12-15	15-18	18-21	21-24	24-27	27-30
ftp	■	■		■	■	■		■	■	
game			■	■			■	■	■	
email										
internet				■		■	■		■	
open office							■	■		■
emacs			■	■		■	■			■
a.out		■	■	■	■	■	■	■	■	■
complie		■	■	■	■	■	■	■	■	■
vi		■				■	■	■	■	■
terminal	■	■	■	■	■	■	■	■	■	■
media player		■	■	■	■	■	■	■	■	■

표 1은 실험에 사용된 시나리오이다. 시나리오 내용은 사용자가 동영상을 보면서 프로그램을 작성하고 있고, 종종 인터넷과 워드프로세스로 보고서를 작성하다가, 대규모의 프로그램을 컴파일 하면서, 동영상을 보며 게임을 하고 있다. 이런 상황에서 시스템에 많은 부하가 생기면 미들웨어는 동영상이 끊기지 않게 컴포넌트 미들웨어 구성을 실시간으로 변경한다.

실험에 이용할 데이터를 얻기 위해서 모니터링 프로그램을 실행하면서 약 1시간 30분 정도 시나리오를 재현하였다. 수집된 데이터로 베이지안 네트워크를 학습시켰고, 학습으로 설계된 베이지안 네트워크 구조의 성능을 파악하기 위해 테스트 데이터를 별도로 약 30분 정도 수집하였다.

3.3 컴포넌트 재구성

추론 결과를 얻기 위해 if-then 규칙을 사용하면, 시스템에 영향을 줄 수 있는 변수와 사용중인 어플리케이션의 복잡한 관계를 모두 설계하기 힘들다. 그러나 베이지안 네트워크는 복잡한 관계표현에 용이하고, 유연한 추론이 가능하다. 따라서 베이지안 네트워크를 사용하여 추론을 하고, 그 추론결과를 바탕으로 실행단계에 명령할 때는 단순한 몇 개의 경우로 나누어 표현이 가능한 if-then 규칙을 사용하였다.

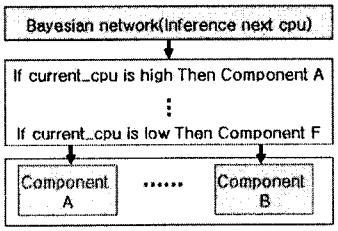

그림 4. if - then 규칙을 이용한 컴포넌트 재구성

4. 실험 및 결과

4.1 실험환경

표 2는 시나리오 실행 후 모니터 프로그램으로부터 얻은 정보를 토대로 작성한 표이다. 1에서 11까지 숫자로 표기한 것은 실행중인 프로세스에 대한 기록을 남긴 것이다. (숫자1은 ftp, 2는 game, 3은 e-mail, 4는 internet, 5는 word processor, 6은 editor, 7은 a.out file, 8은 compile, 9는 vi editor, 10은 terminal, 11은 media player를 의미하고 숫자 0은 해당 프로세스를 사용하지 않을 때를 보여준다. a.out file은 while문을 이용한 무한루프 프로그램을 컴파일하여 실행시킨 것이다.) 가장 우측 실수로 표기된 숫자는 CPU상태를 나타낸 것이다.

표 2. 시스템과 프로세스에 대한 데이터

Mon Aug 29 19:40:22	1	2	0	0	0	0	7	8	0	10	11	7.31
Mon Aug 29 19:40:33	1	2	0	0	0	0	7	8	0	10	11	7.5
Mon Aug 29 19:40:44	1	2	0	0	0	0	7	8	0	10	11	7.65
Mon Aug 29 19:40:55	1	2	0	0	0	0	7	8	0	10	11	7.64
Mon Aug 29 19:41:07	1	2	0	0	0	0	7	8	0	10	11	8.15
Mon Aug 29 19:41:17	1	2	0	0	0	0	7	8	0	10	11	8.36
Mon Aug 29 19:41:30	1	2	0	4	0	0	7	8	0	10	11	8.51
Mon Aug 29 19:41:45	1	2	0	4	0	0	7	8	0	10	11	9.76
Mon Aug 29 19:41:58	1	2	0	4	0	0	7	8	0	10	11	9.58
Mon Aug 29 19:42:13	1	2	0	4	0	0	7	8	0	10	11	11.08
Mon Aug 29 19:42:32	1	2	0	4	0	0	7	8	0	10	11	14.04
Mon Aug 29 19:42:45	1	2	0	4	0	0	7	8	0	10	11	14.5

모니터 프로그램에서 얻은 데이터 분석 결과 CPU상태가 8.6이상이면 시스템에 부하가 생겨서 동영상이 끊어졌다.

그림 5는 데이터에서 얻은 CPU상태 그래프로서 실선을 넘을 경우 동영상이 끊기는 현상을 보였다. 실선은 임계치를 표현하며 약 8.6의 값을 가진다.

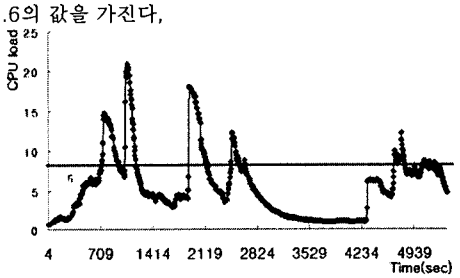

그림 5. cpu상태 그래프

베이지안 네트워크는 현재 사용자가 사용하는 프로그램 목록과 현재 CPU상태를 토대로 3초 후의 상태를 추론하도록 하였다.

4.2 실험 결과

그림 6은 학습한 베이지안 네트워크 구조이다. 베이지안 네트워크 학습 시, 조건부 확률테이블 값도 같이 생성했으며, 베이지안 네트워크 구조의 정확성을 확인하기 위해 별도로 수집했던 30분 분량의 데이터를 사용하여 추론했다. 그 결과, 학습 데이터에 대해 86.5%의 정확성을 보였다.

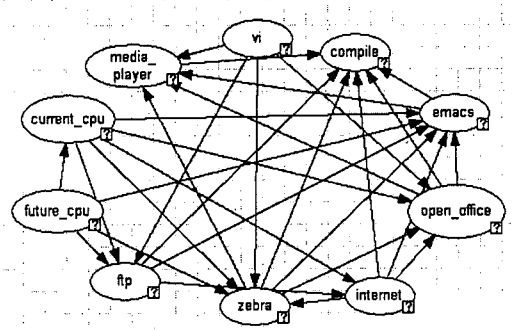

그림 6. 학습한 베이지안 네트워크 구조(current_cpu와 future_cpu는 8.6이상일 때 참이고 이하일 때 거짓이다.)

테스트 데이터로 실험한 결과, 컴포넌트를 바꿔주어야 하는 상황은 세 차례 발생하였지만, 추론에서는 네 차례 예상하여 76.5%의 정확성을 보였다. 이는 실질적으로 컴포넌트 교체가 일어나야 하는 상황을 모두 예상 하였고, 컴포넌트 교체가 일어나야 할 정도의 불안정한 상황도 예측한 것이기 때문에 안정적인 결과이다.

표 3. 성능표

	CPU LOAD>8.6	CPU LOAD<8.6	Accuracy
CPU LOAD>8.6	25	5	83.3%
CPU LOAD<8.6	129	412	76.1%
Total	154	417	76.5%

5. 결론 및 향후 연구

본 논문에서 컴포넌트 기반 미들웨어가 스스로 상황을 판단하여 안정적인 시스템을 유지하도록 베이지안 네트워크를 사용한 컴포넌트 재구성 시스템을 구현해 보았다. 베이지안 네트워크 설계와 조건부 확률 테이블의 값 설정은 학습을 통해 이루어졌다. 실험 결과 학습 데이터에 대해 86.5%의 정확성을 보였고, 테스트 데이터에 대해 76.5%의 정확성을 확인했다.

향후 연구로는 베이지안 네트워크가 더 많은 증거노드를 확보할 수 있도록 시스템 환경에 영향을 줄 수 있는 데이터에 대하여 광범위하게 조사하고, 증거노드에 증거 값이 누락된 경우에 추론 가능성을 실험해 볼 필요가 있다.

참고 문헌

[1] A. Diaconescu and J. Murphy, "A framework for using component redundancy for self-optimizing and self-healing component based systems," *19th European Conf. on Object-Oriented Programming*, pp. 25-29, 2005.

[2] D. Heckerman, "A tutorial on learning with Bayesian networks," *Technical Report. MSR-TR-95-06*, Microsoft Research, 1995.