J. Soc. Korea Ind. Syst. Eng Vol. 38, No. 3 : 56-63, September 2015
http://dx.doi.org/10.11627/jkise.2015.38.3.56
ISSN : 2005-0461(print)
ISSN : 2287-7975(online)

Runtime Prediction Based on Workload-Aware Clustering

Eunhye Kim* · Ju-Won Park**†

*IT Convergence Technology Research Laboratory, ETRI
**Korea Institute of Science and Technology Information, KISTI

병렬 프로그램 로그 군집화 기반 작업 실행 시간 예측모형 연구

김은혜* · 박주원**†

*한국전자통신연구원 융합기술연구소
**한국과학기술정보연구원 슈퍼컴퓨팅본부

Several fields of science have demanded large-scale workflow support, which requires thousands of CPU cores or more. In order to support such large-scale scientific workflows, high capacity parallel systems such as supercomputers are widely used. In order to increase the utilization of these systems, most schedulers use backfilling policy: Small jobs are moved ahead to fill in holes in the schedule when large jobs do not delay. Since an estimate of the runtime is necessary for backfilling, most parallel systems use user's estimated runtime. However, it is found to be extremely inaccurate because users overestimate their jobs. Therefore, in this paper, we propose a novel system for the runtime prediction based on workload-aware clustering with the goal of improving prediction performance. The proposed method for runtime prediction of parallel applications consists of three main phases. First, a feature selection based on factor analysis is performed to identify important input features. Then, it performs a clustering analysis of history data based on self-organizing map which is followed by hierarchical clustering for finding the clustering boundaries from the weight vectors. Finally, prediction models are constructed using support vector regression with the clustered workload data. Multiple prediction models for each clustered data pattern can reduce the error rate compared with a single model for the whole data pattern. In the experiments, we use workload logs on parallel systems (i.e., iPSC, LANL-CM5, SDSC-Par95, SDSC-Par96, and CTC-SP2) to evaluate the effectiveness of our approach. Comparing with other techniques, experimental results show that the proposed method improves the accuracy up to 69.08%.

Keywords : Runtime Predictor, Parallel Workload, Support Vector Regression, Clustering Analysis

1. 서 론

전통적으로 고에너지 물리, 해양, 기상, 천문 우주 등 많은 과학 분야에서 다수의 코어를 동시에 활용하는 대규모의 워크플로우 지원을 요구하고 있다[4, 11]. 이러한 대규모의 워크플로우를 지원하기 위해 슈퍼컴퓨터와 같은 대용량 컴퓨터가 활용된다. 슈퍼컴퓨터에서 가장 기본이 되는 스케줄링 정책은 FCFS(First Come First Service)이다. FCFS 서비스 정책은 매우 간단하고 보편적으로 많이 활용되고 있지만 파편화된 자원 사용으로 인해 작업에 할당되지 않고 유휴 상태에 있는 자원이 많아 시스템 사용률이 매우 낮다. 이러한 문제를 해결하고 시스템 사용률을 높이기 위해서 backfilling 방식이 많이 활용된다. Backfilling은 큐에 있는 큰 규모의 작업이 지연되지 않은 범위 내에서 유휴 상태에 있는 자원을 큐 뒤쪽에 있는 소규모 작업에 할당하여 먼저 실행 하도록 허용 하는 방법이다[16, 18,

Received 5 June 2015; Finally Revised 26 June 2015;
Accepted 4 September 2015
† Corresponding Author : juwon.park@kisti.re.kr

22]. 많은 연구에서 backfilling 방식을 통해 자원 사용률을 높일 수 있음이 입증되어 현재 많은 슈퍼컴퓨터 센터에서 활용하고 있다. 예를 들어 스웨덴의 HPC2N(High Performance Computing Center North), 미국의 NCSA(National Center for Supercomputing Applications), SDSC(San Diego Supercomputer Center) 등에서 활용되고 있으며 KISTI의 타키온에서도 Sun Grid Engine(SGE)에 backfilling 기법이 적용되어 서비스되고 있다. 이러한 backfilling 기법을 이용하기 위해서는 작업 실행 시간을 예측하는 것이 필수적이다.

병렬 프로그램 작업 시간 예측을 위해 대부분의 슈퍼컴퓨터에서는 사용자가 예측한 작업시간을 기반으로 backfilling 기법이 적용되고 있다. 그러나 사용자는 자신의 작업이 완료되기 전 스케줄러에 의해 강제로 종료되는 것을 방지하기 위해 실제 실행 시간보다 더 큰 값으로 예측한다[18]. 또한 많은 경우 대부분 30분 또는 1시간 단위로 제시함으로써 실제 작업이 실행되는 시간과 큰 차이가 존재한다. 이러한 문제를 해결하고 예측의 정확도를 높이기 위해 시스템에서 실행 시간을 예측하고자 하는 많은 연구가 진행되었다[6, 10, 17, 20, 21]. 그러나 작업의 실행 시간 변화량이 매우 크고 동일한 작업이라 할지라도 실행되는 시스템, 입력 파라미터에 따라 실행 시간이 매우 다양하기 때문에 예측에 많은 어려움이 있다.

본 논문에서는 모든 시스템에 동일한 모델을 기반으로 작업 실행 시간을 예측하는 것이 아닌 각 시스템에서 수집된 과거 로그 이력 데이터를 이용하여 시스템에 따른 적합한 모델을 만들고 이를 기반으로 새로운 작업의 실행 시간을 예측하도록 한다. 제안된 방식은 크게 3단계로 구분할 수 있다. 먼저, 요인분석(factor analysis)을 이용하여 시스템별 수집된 과거 로그 이력 데이터의 주요 속성을 분석한다. 둘째, 요인 분석 결과를 기반으로 비지도 학습(unsupervised learning) 기법인 Self-Organizing Map (SOM)과 계층적 군집분석(hierarchical clustering) 기법[13]을 이용하여 병렬 프로그램의 작업 군집화를 수행한다. 마지막으로, 병렬 프로그램 로그 데이터의 유사 패턴별로 Support Vector Regression(SVR)을 이용한 예측 모델을 구축하여 작업 실행 시간을 예측한다. 제안 기법의 성능 평가를 위하여 표준화된 워크로드 형식(Standard Workload Format)[8]으로 저장된 NASA iPSC, LANL-CM5, SDSC-Par95, SDSC-Par96, CTC-SP2 로그 데이터를 이용한 실험을 수행한다.

본 연구의 제 2장에서는 병렬 프로그램 작업 시간 예측 관련 연구를 살펴본다. 제 3장에서는 병렬 프로그램 작업 군집화 기반 예측 모델 구축에 대한 제안 기법을 설명한다. 제 4장에서는 제안한 작업 실행 시간 예측 모델의 성능 비교분석 결과를 기술하고, 마지막으로 제 5장에서 결론을 맺는다.

2. 관련 연구

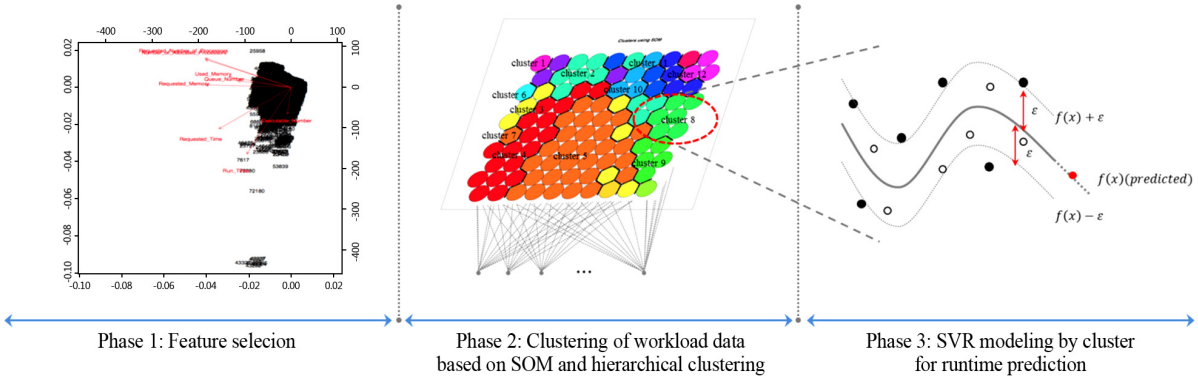
일반적으로 병렬 프로그램을 이용하는 사용자는 동일한 응용 프로그램을 반복적으로 사용하는 것으로 알려져 있기 때문에 워크로드가 반복된 형태를 지니는 것으로 알려져 있다[5, 9]. 이러한 특징으로 인해 과거 로그 이력 데이터를 기반으로 응용 프로그램의 실행 시간을 예측하고자 하는 많은 연구가 진행되었다. 이러한 관련 연구는 유사한 작업을 찾는 방법에 따라 다음 3가지로 나누어 볼 수 있다.

- Templet 기반 예측 기법 : Templet 기반 방식은 작업의 속성에 따라 구분된 templet으로 작업의 유사도를 판단하여 예측하는 방법이다. Downey[6]가 제안한 기법은 시스템 큐(queue)값을 이용하여 templet을 정의하고 클래스를 할당하여 동일한 클래스에 속하는 작업을 기반으로 실행 시간을 예측하였다. Gibbons[10]는 사용자 ID나 시스템 큐와 같이 정의된 속성을 기반으로 응용 프로파일러(application profiler)를 정적(static)으로 정의하여 작업을 분류하였다. Smith et al.[21]은 유사도를 특정 속성으로 정의하는 것이 아닌 예측의 정확도를 높이기 위해 greedy와 genetic algorithm을 이용하여 동적으로 적합한 유사도 결정하였다.
- Instance 기반 예측 기법 : Instance 기반 방식은 과거 로그 이력 데이터를 분류하지 않고 예측하고자 하는 작업과 유사한 k개의 작업을 찾아 이를 기반으로 실행 시간을 예측한다. Senger et al.[20]이 제안한 기법은 과거 로그 이력 데이터의 모든 작업을 n개의 차원공간으로 매핑하여 예측하고자 작업과 가까운 k개의 근접 데이터를 기반으로 실행 시간을 예측한다.
- Smoothing Method 기반 예측 기법 : Tsafrir et al.[22]이 제시한 기법으로 동일한 사용자가 제출한 최근 2개의 실행 시간 평균으로 실행 시간을 예측하는 방법이다. 본 방식은 알고리즘이 매우 간단하여 쉽게 구현할 수 있는 장점이 있다.

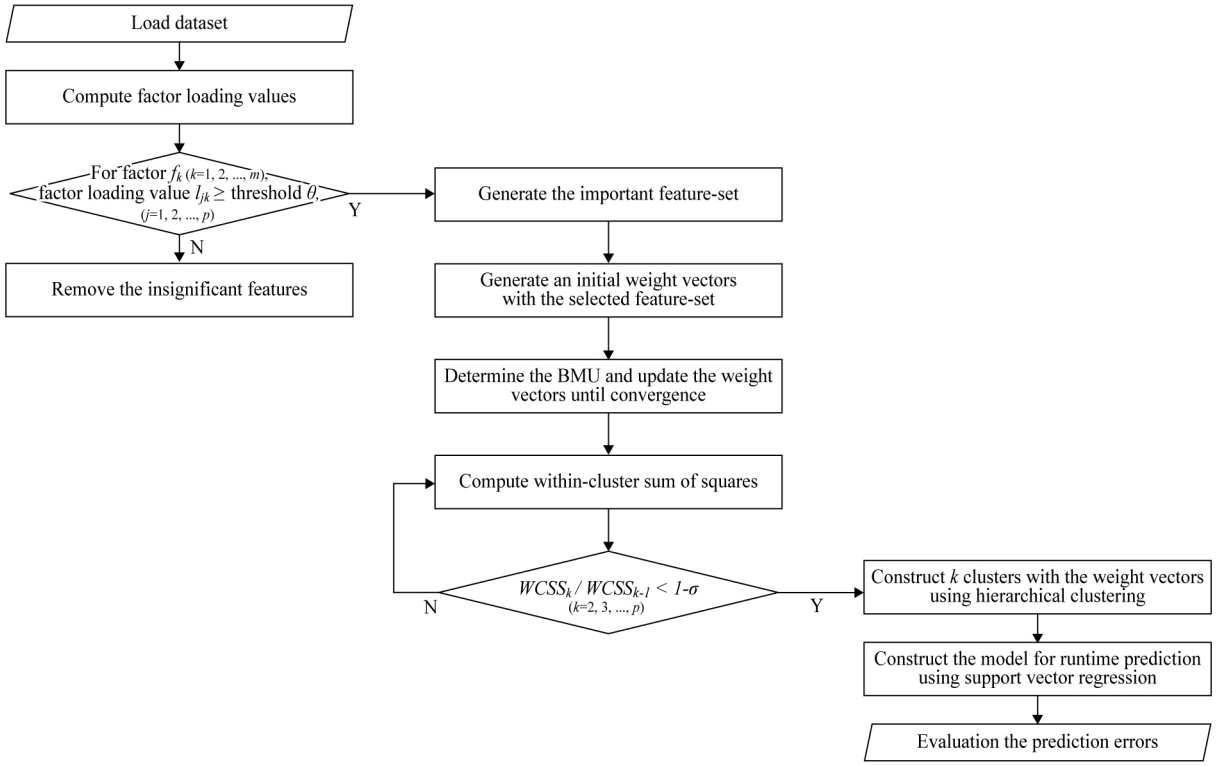
Templet 기반 예측 기법과 smoothing method 기반 예측 기법은 동일한 속성을 가지는 데이터가 과거 로그 이력 데이터에 존재할 경우에만 예측이 가능하다. 또한 병렬 프로그램 로그 데이터의 경우 동일한 속성을 가지는 중복 데이터 수가 많아 k-최근접 이웃(k-nearest neighbors) 알고리즘만을 이용한 작업 실행 시간 예측 방식은 오차가 매우 크다는 단점이 있다. 그러나 본 연구의 제안 기법은 동일한 속성의 과거 로그 데이터가 존재하지 않는 경우에도 자율 학습 방식의 클러스터링 기법을 기반으로 유사 군집을 찾아 실행시간 예측이 가능하며, 군집별 예측모형 구축으로 모델의 실행시간 예측력이 강화된다는 장점이 있다.

3. 제안 기법

병렬 프로그램의 실행 시간 예측성능 향상을 위해 본 연구에서 제안한 기법은 <Figure 1>과 같이 3단계로 구분할 수 있다. 첫 번째 단계는 작업의 중요 속성을 추출하는 단계이다. 이는 변수들 간의 구조적 관계를 해석하는 통계기법인 요인분석을 통해 유인구조를 파악하고 요인적재값(factor loading)을 분석하여 작업의 중요 속성을 선택한다. 두 번째 단계는 SOM과 계층적 군집분석 기법을 이용한 작업군집화 단계이다. 추출된 중요 특징으로 구성된 과거 로그 이력 데이터에 SOM의 경쟁 학습을 이용하여 가중치벡터를 계산하고, 최적 군집수를 결정하기 위해 가중치벡터의 군집별로 within-cluster sum of squares(WCSS)를 계산한다. 군집수에 따른 WCSS의 증감률이 임계치(threshold)를 초과하는 WCSS 값을 기반으로 최적 군집수를 결정하여 군집화한다. 마지막 단계는 가중치벡터의 군집화를 통한 병렬 프로그램 로그 데이터의 유사 패턴별로 Support Vector Regression(SVR)을 이용한 예측모델을 구축하여 작업 실행시간을 예측하며, 평균제곱근오차(Root Mean Square Error)를 이용하여 구축 모델의 예측력을 평가한다. 제안한 기법의 절차는 <Figure 2>에서 보는 바와 같다.



<Figure 1> Overview of the Proposed Framework



<Figure 2> Proposed Runtime Prediction Procedure

3.1 Phase 1 : Feature Selection

요인분석은 다변량분석의 대표적 기법 중 하나로, 변수들 간의 상관관계를 분석하고 공통적으로 작용하는 내재적 특성을 추출하여 전체자료를 표현할 수 있는 변수의 수를 축소하는 통계적 기법이다[12]. 변수 $\boldsymbol{x}=(x_1, x_2, \cdots, x_p)^T$에 대하여 $m(m<<p)$ 개의 요인을 고려하는 통계적 모형을 수리적으로 표현하면 아래 수식과 같다.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} l_{11}\, l_{12} & \cdots & l_{1m} \\ l_{21}\, l_{22} & \cdots & l_{2m} \\ & \vdots & \\ l_{p1}\, l_{p2} & \cdots & l_{pm} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_p \end{bmatrix}$$

여기서, 확률변수 $\boldsymbol{f}=(f_1, f_2, \cdots, f_m)^T$는 모든 측정변수들의 변이에 영향을 주는 공통요인(common factor)이고, $\boldsymbol{\epsilon}=(\epsilon_1, \epsilon_2, \cdots, \epsilon_p)^T$는 각 측정변수에 해당하는 오차로 유일성 변수이다. 위의 식에서 선형결합에 사용된 가중계수 l_{ij}는 요인적재값으로, 모형에서 고려된 i번째 변수 x_i에 관한 j번째 요인 f_j의 중요성을 나타낸다. 요인분석은 결국 공분산행렬로부터 요인적재값 l_{ij}와 유일성을 추정하는 것으로, 고유값(eigenvalue) γ_i와 고유벡터(eigenvector) e_i로 표현되는 공분산행렬로부터 추정되는 요인적재값 $\{\hat{l}_{ij}\}$은 아래 수식과 같다.

$$\hat{\mathbf{L}} = [\sqrt{\gamma_1}e_1,\ \sqrt{\gamma_2}e_2, \cdots,\ \sqrt{\gamma_m}e_m]$$

요인적재값은 변수와 요인 간의 상관계수로 요인적재값의 제곱은 해당 변수가 요인에 의하여 설명되는 분산의 비율을 나타낸다. 본 연구에서는 요인분석을 통해 유인구조를 파악하고 요인적재값을 분석하여 작업 로그 데이터의 주요 속성을 추출하였다.

3.2 Phase 2 : Clustering of Workload Data

SOM은 학습단계에서 다차원 공간 상의 특징벡터로 표현되는 객체들이 경쟁층에서 스스로 유사한 패턴끼리 이차원의 특징 지도를 조직화하는 비지도 학습 알고리즘이다[14, 15]. 군집화의 결과는 특징 지도로 나타나며 SOM의 학습은 경쟁학습을 기초로 한다. 개체벡터에 가장 가까운 가중치벡터를 찾아 개체벡터 방향으로 이동시키는 과정을 반복하는 데, 이 과정에서 가장 가까운 가중치벡터의 주변 가중치벡터도 함께 개체벡터 방향으로 이동시킨다. 여기서 가장 가까운 가중치벡터는 승자노드(best matching unit)라고 부른다. 학습의 반복은 모든 가중치벡터 값의 변화가 거의 소멸하거나 미리 지정된 최대 한계에 도달할 때까지 계속 진행되며, 이러한 학습과정을 통해 얻은 최종적인 가중치벡터는 개체공간상에서 개체들이 지니는 위상적 지도를 반영하게 된다. <Table 1>은 본 논문에서 사용할 SOM 수행 절차 알고리즘을 나타낸 것이다.

SOM은 학습단계에서 피드백 과정이 없어 구조적으로 상당히 학습수행이 빠르다는 장점을 가지로 있으나, 변수의 크기가 증가할수록 군집화의 성능이 떨어지는 단점을 보인다[2]. 이러한 단점을 보안하고 군집화 성능을 높이기 위해 본 연구에서는 요인분석을 통해 추출된 주요 특징변수만을 이용하여 속성이 유사한 작업데이터를 군집화한다.

<Table 1> Learning Procedure of the SOM

Step 1	Construct a weight vector, w, and initialize it
Step 2	Calculate the Euclidean distance between the ith input vector x_i and each unit weight w_j, and choose the winning unit or BMU, $d=\min\|x_i(t)-w_j(t)\|$
Step 3	Adjust the weights for the winners and all its neighbors as $w_{ij}(t+1)=w_{ij}(t)+\alpha(t)h(j,t)(x_i(t)-w_{ij}(t))$, where $\alpha(t)$ is the learning rate at epoch t and $h(j,t)$ is a the neighborhood kernel function centered on the winning unit, $h(j,t)=\exp\left(-\frac{d_{ij}^2}{2\sigma^2(t)}\right)$
Step 4	Repeat steps (2)~(4) until the convergence criterion is satisfied

본 단계에서는 SOM의 경쟁학습을 이용하여 가중치벡터를 계산하고, 최적 군집수를 결정하기 위해 가중치벡터의 군집별로 within-cluster sum of squares(WCSS)를 계산한다. 군집수에 따른 WCSS의 증감률이 임계치(threshold)를 초과하는 WCSS 값에 해당하는 군집수 중 최대치를 최적 클래스 수로 결정한다. 본 연구의 WCSS 비는 아래와 같이 정의한다.

$$\sum_{i=1}^{k}\sum_{x\in S_i}\|x-\mu_{S_i}\|^2 / \sum_{i=1}^{k-1}\sum_{x\in S_i}\|x-\mu_{S_i}\|^2 < 1-\sigma$$

위 식에서 x는 입력벡터, S는 k개로 분할된 가중치벡터 집합, μ_{S_i}는 i번째 집합의 중심점을 의미한다. WCSS 비를 통해 결정된 최적 군집수와 계층적 군집분석을 이용하여 가중치벡터를 군집화함으로써 병렬 프로그램 로그 데이터의 유사 패턴을 분석한다.

3.3 Phase 3 : SVR Modeling by Clusters

Vapnik[23]에 의해 제안된 SVM은 구조적 위험 최소화(Structural Risk Minimization)에 기초하여 입력공간과 관련된 비선형 문제를 고차원의 특징공간의 선형문제로 대응시키는 학습 알고리즘으로, 경험적 위험 최소화(Empirical Risk Minimization)에 기초한 기법들에 비해 우수한 성능을 보이는 기계학습기법으로 주목 받고 있으며, 분류, 인식, 이상탐지 등에서 성공적으로 적용되어 왔다[1, 3, 24]. SVM은 학습 데이터의 분류문제의 예측에 사용되지만, 임의의 실수값을 예측할 수 있도록 SVM 모형에 ε-무감도 손실함수(ε-insensitive loss function)를 도입한 Support Vector Regression(SVR)이 회귀문제의 영역까지 확장되었다[7]. SVR은 입력값의 비선형 사상을 통해 목표 변수를 가장 잘 설명하는 다음과 같은 회귀함수를 추정한다.

$$f(x) = w \cdot \Phi(x) + b$$

즉, 비선형회귀를 이용한 원래의 최적화 문제는 형상공간에서 선형함수를 탐색하는 문제로 재정의된다. ε-무감도 손실함수 L_ϵ는 일반적으로 SVR에 사용되는 비용함수로 다음과 같다.

$$L_\epsilon(f(x), q) = \max(0, |f(x) - q| - \varepsilon)$$

여기서 ε은 회귀함수 $f(x)$의 주변에 위치한 튜브의 반지름을 나타내는 정밀모수(precision parameter)이다. SVR에서는 실제값과 예측값을 가능한 한 ε 이내로 유지하면서 마진을 최대화한다. 학습과정에서 오류를 허용하는 여유변수와 추정함수의 완만함과 학습오차간의 상충관계를 조절하는 상수를 이용한 SVR의 최적화 문제에 Lagrangian 승수와 Karush-Kuhn-Tucker 조건을 적용하면, SVR 기반 회귀 함수의 일반적 형태가 다음과 같이 도출된다.

$$f(x, \alpha, \alpha^*) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + b$$

위의 식에서 $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$는 커널함수로 본 논문에서는 다음과 같은 Radial Basis Function 커널 함수를 사용하였으며,

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

본 논문에서는 작업시간 예측력 향상을 위해 SOM과 계층적 군집분석을 통해 분석된 속성이 유사한 작업데이터의 군집별로 SVR 모형을 구축하여 작업 실행 시간을 예측한다.

4. 실험 결과

4.1 실험 환경

본 실험에서는 <Table 2>에서 제시한 바와 같이 Feitelson [8]에서 제공하는 병렬 워크로드 로그 파일을 사용하여 병렬 프로그램의 작업 시간을 예측하고 제안된 방식의 예측력을 평가한다. Feitelson[8]에서는 다수의 슈퍼컴퓨팅 센터에서 과거 로그 이력 데이터를 수집하여 표준화된 워크로드 형식[5]으로 정리하여 제공하고 있으며 아래와 같은 18개의 속성으로 구성된다.

<Table 2> Parallel Workload Logs.

Workload log	System	Duration (in month)	Version
NASA-iPSC	128-node iPSC/ 860 hypercube	3	3.1 cleaned
LANL-CM5	1024-node CM5	24	3.1 cleaned
SDSC-Par95	416-node Intel Paragon	12	3.1 cleaned
SDSC-Par96	416-node Intel Paragon	12	3.1 cleaned
CTC-SP2	512-node IBM SP2	11	3.1 cleaned

- Job Number : 정수로 표기된 작업번호. 로그 파일에 기록된 첫 번째 작업을 1로 하여 순차적으로 부여.
- Submit Time : 초단위로 기록된 작업 제출 시각. 로그 파일에 기록된 1번 작업의 제출 시간을 0으로 하여 제출된 시간의 차이를 기록.
- Wait Time : 초단위로 기록된 큐 대기 시간. 작업 제출 시각과 실제 작업이 실행한 시각의 차이로 기록.
- Run Time : 초단위로 기록된 작업 실행 시간. 작업이 시작된 시각과 종료된 시각의 차이로 기록됨. 본 논문에서 예측하고자 하는 종속변수.
- Number of Allocated Processors : 작업에 할당한 프로세서의 수.
- Average CPU Time Used : 초단위로 기록된 CPU 사용시간.
- Used Memory : KB 단위로 기록된 메모리 사용량.
- Requested Number of Processors : 사용자에 의해 요청된 프로세스 수.
- Requested Time : 초단위로 기록된 사용자가 예측한

작업 실행 시간.

- Sequested Memory : KB 단위로 기록된 사용자에 의해 요청된 메모리 용량.
- tatus : 정수로 기록된 작업 상태 코드(0~5 값으로 표기).
- User ID : 사용자 아이디.
- Group ID : 그룹 아이디.
- Executable (Application) Number : 동일한 워크로드에 있는 다수의 응용 프로그램을 구분하기 위해 기록된 값.
- Queue Number : 하나의 시스템에 다수의 큐가 존재할 경우 이를 구분하기 위해 기록된 값.
- Partition Number : 하나의 시스템에 다수의 파티션(partition)이 존재할 경우 이를 구분하기 위해 기록된 값.
- Preceding Job Number : 본 작업이 시작하기 전에 종료되어야 하는 이전 작업.
- Think Time from Preceding Job : 초단위로 기록된 값으로 이전 작업이 종료되고 제출된 작업이 시작 할 때까지의 시간.

본 논문에서는 각 시스템에서 얻어진 로그 파일에서 상태 코드 값이 1(정상 종료된 작업)인 작업을 대상으로 실험을 수행하였다. 로그 분석을 위해 대표적인 통계 프로그래밍 언어 중의 하나인 R[19]을 사용하였으며 kohonen, e1071, party, kknn 패키지를 이용하였다. 모델 구축 및 평가에 사용한 로그 데이터는 <Table 3>과 같다.

<Table 3> Data Sets

Workload log	Records	Ratio of training data	Ratio of test data
NASA-iPSC	17,195	70.9%	29.1%
LANL-CM5	76,697	93.5%	6.5%
SDSC-Par95	53,064	90.6%	9.4%
SDSC-Par96	31,321	84.0%	16.0%
CTC-SP2	60,548	91.7%	8.3%

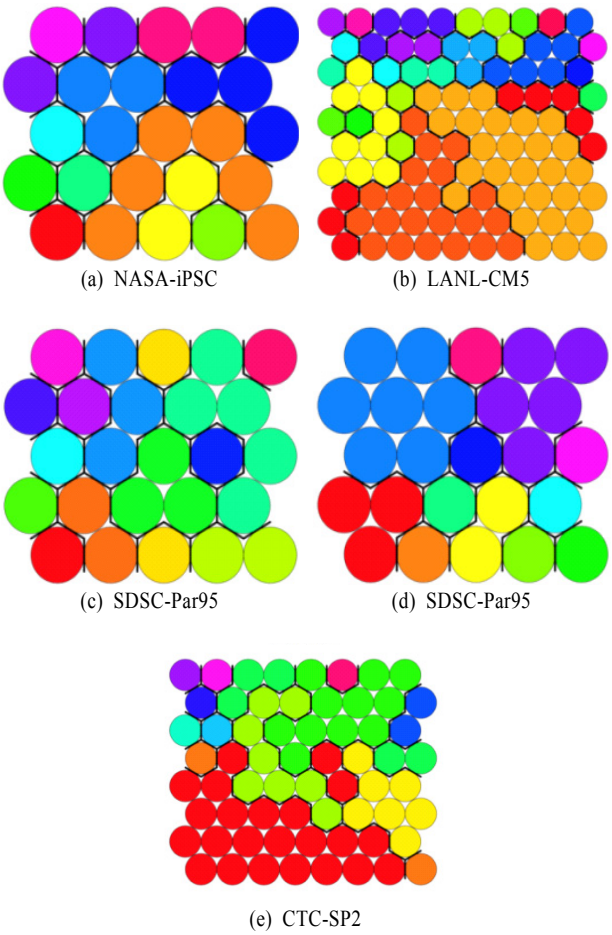
<Table 4> Important Features

Workload Log	Important Features
NASA-iPSC	Number of Allocated Processors, Executable Number
LANL-CM5	Number of Allocated Processors, Requested Number of Processors, Requested Time, Requested Memory
SDSC-Par95	Number of Allocated Processors, Queue Number, Partition Number,
SDSC-Par96	Number of Allocated Processors, Queue Number, Partition Number
CTC-SP2	Number of Allocated Processors, Requested Time, Executable Number, Queue Number

4.2 실험 결과

<Table 4>는 로그 데이터에서 값이 존재하지 않는 속성과 'Average CPU Time Used' 속성과 같이 작업이 완료된 후 측정이 가능한 속성을 제외한 후 요인 분석을 통한 요인적재값을 기반으로 중요 속성을 추출한 결과이다.

각 시스템의 주요 속성을 기반으로 SOM 학습결과 얻어진 U-Matrix와 군집화 결과는 <Figure 3>와 같다. U-Matrix는 가중치 벡터를 이용하여 이웃한 셀 간의 거리를 계산하여 2차원 공간상에 표현한 것으로, <Figure 3>의 각 셀의 색과 굵은 선으로 표시한 경계선은 WCSS 비를 통해 결정된 최적 군집수를 이용한 가중치벡터의 군집화 결과를 보여준다. 군집수에 따른 WCSS의 증감률이 임계치를 초과하는 WCSS에 해당하는 군집수 중 최대치를 최적 클래스 수로 결정함에 따라 NASA-iPSC 로그 데이터는 12개, LANL-CM5는 18개, SDSC-Par95는 14개, SDSC-Par96은 12개, CTC-SP2는 13개의 군집으로 구분되었다.



(a) NASA-iPSC (b) LANL-CM5

(c) SDSC-Par95 (d) SDSC-Par95

(e) CTC-SP2

<Figure 3> SOM Clustering of Workload Logs

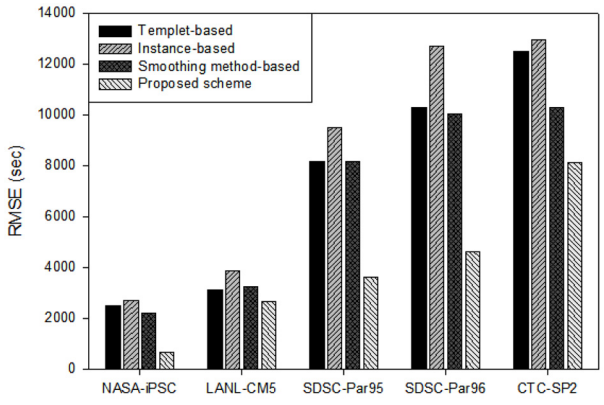
결정된 군집수에 따라 계층적 군집분석을 이용하여 가중치벡터를 군집화하고 속성이 유사한 작업데이터의 군집별로 SVR 모형을 구축하여 작업 실행 시간을 예측하였다. 모델의 예측력 평가를 위해 Root Mean Square Error (RMSE)을 이용하였다.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

여기서 y_i는 실측값, $\hat{y}_i$는 예측값, n은 데이터 수를 의미한다.

<Figure 4>는 제안된 기법과 기존 연구에서 수행된 templet, instance, smoothing method 기반의 예측 성능을 비교한 결과를 보여준다. Templet 기반 예측으로는 smith[21], instance 기반 예측 기법으로는 k-nearest neighbors algorithm[20], smoothing method 기반 예측 기법으로는 Tsafrir [22]의 모델을 각각 사용하였다.

<Figure 4>의 결과를 통해 확인할 수 있듯이 본 연구에서 제안한 군집화 기반 SVR 기법이 실험에서 사용된 모든 시스템 로그에서 다른 비교 모형에 비해 우수한 예측 정확도를 보이고 있다. 특히, NASA-iPSC 데이터에 대한 비교 군에서는 Smoothing Method 기반 예측 모델이 2198.09초의 오차로 우수한 성능을 보였으나 본 연구의 제안 기법의 예측 오차는 679.63초로, Smoothing Method 기반 예측 모델 대비 69.08%의 성능 향상을 보였다. LANL-CM 로그의 비교 군 실험에서는 templet 기반 예측 기법이 3114.93초의 오차로 가장 우수한 성능을 보였으며 제안된 방식은 2666.06초로, templet 기반 예측 모델 대비 14.41%의 성능 향상을 보였다. SDSC-Par95, SDSC-Par96, CTC-SP2 비교 군에서는 Smoothing Method 기반 예측 기법의 성능이 우수하였으며 본 연구의 제안 기법은 Smoothing Method 기반 예측 모델 대비 각각 55.84%, 53.82%, 20.83%의 성능 향상을 보였다.



<Figure 4> Comparison of Performance Results

5. 결 론

대부분의 슈퍼컴퓨터에서 사용하는 backfilling 기법을 이용하기 위해서는 작업 실행 시간 예측 값이 필수적이다. 이를 위해서 많은 슈퍼컴퓨팅 센터에서 사용자가 예측하여 입력한 값을 기반으로 스케줄링을 실행하고 있지만 이는 실제 작업 시간과 많은 차이가 발생한다. 이를 위해 본 논문에서는 과거 로그 이력 데이터를 기반으로 기계 학습 기법을 통해 병렬 프로그램실행 시간을 예측하는 방안을 제시하였다. 제시된 기법의 예측 성능을 분석하기 위해 NASA Ames Research Center, Los Alamos National Lab(LANL), San Diego Supercomputer Center(SDSC), Cornell Theory Center(CTC)에서 수집된 로그 정보를 활용하였으며, RMSE를 이용한 모형별 성능 분석 결과 본 연구의 제안 기법은 관련 연구에서 제시된 기법들 대비 최소 14.41%에서 최대 69.08%의 성능 향상을 보였다. 이는 대용량 병렬 컴퓨팅 자원의 효율 향상을 위한 작업 스케줄링에 있어 제안한 군집화 기반 SVR 기법이 상당히 효과적인 모델이 될 수 있다는 점을 뒷받침하는 연구결과라 할 수 있다.

References

[1] Agarwal, B. and Mittal, N., Text classification using machine learning methods-A survey. *Advances in Intelligent Systems and Computing*, 2014, Vol. 236, pp. 701-709.

[2] Andrews, N.O. and Fox, E.A., Recent developments in document clustering, Technical report, TR-07-35, Department of Computer Science, Virginia Tech, 2007.

[3] Chandola, V., Banerjee, A., and Kumar, V., Anomaly detection : A survey. *ACM Computing Surveys*, 2009, Vol. 41, No. 3, pp. 15-57.

[4] Deelman, E., Gannon, D., Shields, M., and Taylor, I., Workflows and e-science : An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 2009, Vol. 25, No. 5, pp. 528-540.

[5] Downey, A.B. and Feitelson, D.G., "The elusive goal of workload characterization. *ACM SIGMETRICS Performance Evaluation Review*, 1999, Vol. 26, pp. 14-29.

[6] Downey, A.B., Using queue time predictions for processor allocation, in Proc. of the Workshop on Job Scheduling Strategies for Parallel Processing. *Lecture Notes in Computer Science*, 1997, Vol. 1291, pp. 35-57.

[7] Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., and Vapnik, V., Support vector regression machines. *Neural Information Processing Systems*, 1997, pp. 155-

161, MIT Press.

[8] Feitelson, D., Parallel workloads archive and standard workload format, [Online]. Available : http://www.cs.huji.ac.il/labs/parallel/workload.

[9] Feitelson, D.G. and Nitzberg, B., Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860, in Proc. of the Workshop on Job Scheduling Strategies for Parallel Processing. *Lecture Notes in Computer Science*, 1995, Vol. 949, pp. 337-360.

[10] Gibbons, R., A historical application profiler for use by parallel schedulers. *Lecture Notes on Computer Science*, 1997, Vol. 1297, pp. 58-75.

[11] Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., and Myers, J., Examining the challenges of scientific workflows. *IEEE Computer*, 2007, Vol. 40, No. 12, pp. 24-32.

[12] Hardle, W.K. and Simar, L., Applied Multivariate Statistical Analysis, Springer-Verlag, Berlin, 2012.

[13] Kaufman, L. and Rousseeuw, P.J., Finding Groups in Data : An Introduction to Cluster Analysis, John Wiley and Sons, 2009.

[14] Kohonen, T., Essentials of the self-organizing map. *Neural Networks*, 2013, Vol. 37, pp. 52-65.

[15] Kohonen, T., Self-Organizing Map, Springer-Verlag, Berlin, 2001.

[16] Lifka, D.A., The ANL/IBM scheduling system, in Proc. of the Workshop on Job Scheduling Strategies for Parallel Processing. *Lecture Notes in Computer Science*, 1995, Vol. 949, pp. 295-303.

[17] Minh, T.N. and Wolters, L., Using historical data to predict application runtimes on backfilling parallel systems, in Proc. of 18th *Euromicro Conference on Parallel. Distributed and Network-based Processing*, 2010, pp. 246-252.

[18] Mu'alem, A.W. and Feitelson, D.G., Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 2001, Vol. 12, No. 6.

[19] R Development Core Team, R : A language and environment for statistical computing, [Online]. Available : http://www.r-project.org.

[20] Senger, L.J., Santana, M.J., and Santana, R.H.C., An instance-based learning approach for predicting execution times of parallel applications, in Proc. of *International Information and Telecommunication Technologies Symposium*, 2005, pp. 9-15.

[21] Smith, W., Foster, I., and Taylor, V., Predicting application run times with historical information. *Journal of Parallel and Distributed Computing*, 2004, Vol. 64, No. 9, pp. 1007-1016.

[22] Tsafrir, D., Etsion, Y., and Feitelson, D., Backfilling using system-generated predictions rather than user runtime estimates. *IEEE Transactions on Parallel and Distributed Systems*, 2007, Vol. 18, No. 6.

[23] Vapnik, V., The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.

[24] Weinland, D., Ronfard, R., and Boyer, E., A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 2011, Vol. 115, No. 2, pp. 224-241.

ORCID

Eunhye Kim | http://orcid.org/0000-0003-1388-1583

Ju-Won Park | http://orcid.org/0000-0001-7812-9160