KIPS Trans. Softw. and Data Eng.
Vol.9, No.1 pp.11~16 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2020.9.1.11

Development of Application to Deal with Large Data Using Hadoop for 3D Printer

Kang Eun Lee† · Sungsuk Kim††

ABSTRACT

3D printing is one of the emerging technologies and getting a lot of attention. To do 3D printing, 3D model is first generated, and then converted to G-code which is 3D printer's operations. Facet, which is a small triangle, represents a small surface of 3D model. Depending on the height or precision of the 3D model, the number of facets becomes very large and so the conversion time from 3D model to G-code takes longer. Apach Hadoop is a software framework to support distributed processing for large data set and its application range gets widening. In this paper, Hadoop is used to do the conversion works time-efficient way. 2-phase distributed algorithm is developed first. In the algorithm, all facets are sorted according to its lowest Z-value, divided into N parts, and converted on several nodes independently. The algorithm is implemented in four steps; preprocessing - Map - Shuffling - Reduce of Hadoop. Finally, to show the performance evaluation, Hadoop systems are set up and converts testing 3D model while changing the height or precision.

Keywords : Large Data Processing, 3D Printing, G-code, Hadoop, Facet

하둡을 이용한 3D 프린터용 대용량 데이터 처리 응용 개발

이 강 은† · 김 성 석††

요 약

3D 프린팅은 주목받는 신기술의 하나로 많은 관심을 받고 있다. 3D 프린팅을 하기 위해서는 먼저 3D 모델을 생성한 후, 이를 프린터가 인식할 수 있는 G-code로 변환하여야 한다. 대개 3D 모델은 페이셋이라고 하는 조그만 삼각형으로 면을 표현하는데, 모델의 크기나 정밀도에 따라 페이셋의 개수가 매우 많아져서 변환에 많은 시간이 걸리게 된다. 아파치 하둡은 대용량 데이터의 분산처리를 지원하는 프레임워크로서 그 활용 범위가 넓어지고 있다. 본 논문에서는 3D 모델을 G-code로 변환하는 작업을 효율적으로 수행하기 위해 하둡을 활용하고자 한다. 이를 위해 2단계의 분산 알고리즘을 개발하였다. 이 알고리즘은 여러 페이셋들을 먼저 Z축 값으로 정렬한 후, N등분하여 여러 노드에서 독립적으로 분산처리하도록 되어 있다. 실제 분산처리는 전처리 - 하둡의 Map - Shuffling - Reduce의 4 단계를 거쳐 구현되었다. 최종적으로 성능 평가를 위해 테스트용 3D 모델의 크기와 정밀도에 따른 처리 시간의 효율성을 보였다.

키워드 : 대용량 데이터 처리, 3D 프린팅, G-code, 하둡, 페이셋

1. 서 론

여러 국내외 연구기관에서는 지속적으로 데이터의 중요성을 강조하고 있으며, 이러한 '데이터 경제 시대'를 이끌 수 있는 ICT 신기술도 데이터로부터 나오고 있다. 최근 빅데이터나 인공지능이 대표적이며, 이는 이전과 다른 수준의 대용량 데이터 처리를 기반으로 하고 있다[1, 2]. [3]에 따르면 2020년 이후 발생되는 데이터의 양이 폭발적으로 증가할 것으로 예상하고 있다. 데이터의 양이 커진다는 것은 대용량 데이터의 처리 알고리즘 개발뿐만 아니라 효율적으로 처리할 수 있는 기반 기술도 중요해진다.

3D 프린팅이란 출력하고자 하는 3D 모델에 따라 재료를 층층이 쌓아서 3차원 제품을 만드는 기술이다[4,5]. 즉, 프린팅을 하기 위해서는 전용 소프트웨어를 이용하여 출력할 대상에 대한 3D 모델을 먼저 생성해야 한다. 생성된 모델은 다시 프린터가 인식할 수 있는 명령어인 G-code로 변환하는 과정을 거쳐 프린터로 전달되고, 이후 프린팅이 진행된다. 3D 모델은 3차원 표면을 조그만 평면 단위(본 논문에서는 삼각형으로 표시되는 페이셋(Facet)으로 표현함)로 표현하는데, 3차원 모델의 크기나 정밀도에 따라 그 숫자가 매우 많아진다. 이러한 페이셋들간에는 어떤 특별한 규칙없이 3D 모델 파일에 저장된다. 이에 반해 3차원 프린터는 대개 대상의 바닥(Z축의 min값)에서부터 출력하고 단위 크기만큼 이동한 후 다시 출력하는 과정을 반복한다. 따라서 프린터의 동작을 지정하는 G-code를 생성하기 위해서는 페이셋들을 Z 축값에 따라 적절하게 처리해야 한다.

※ 이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2015R1D1A1A01061173).
※ 이 논문은 2019년도 한국정보처리학회 춘계학술발표대회에서 'Hadoop을 이용한 대용량 데이터 변환'의 제목으로 발표된 논문을 확장한 것임.
† 비 회 원 : 서경대학교 소프트웨어학과 학사과정
†† 종신회원 : 서경대학교 소프트웨어학과 교수
Manuscript Received : July 17, 2019
Accepted : August 18, 2019
* Corresponding Author : Sungsuk Kim(sskim03@skuniv.ac.kr)

아파치 하둡(Apach Hadoop)은 대용량 데이터의 분산 처리 응용을 지원하는 프리웨어 자바 소프트웨어 프레임워크이다[6-8]. 분산처리 시스템인 구글 파일 시스템을 대체할 수 있는 하둡 분산 파일 시스템(HDFS: Hadoop Distributed File System)과 맵리듀스(MapReduce)를 구현한 것이다. HDFS는 구글의 GFS기반의 파일 시스템으로서 기존 파일 시스템 위에서 파일을 블록 단위로 분할하여 분산 저장을 지원한다. 이를 위해 Master Node(Job tracker, Name node 역할)와 Slave Node(Data node, Task node 역할)로 구분되어 구동한다. 맵리듀스는 여러 노드에 태스크를 분배하는 방법이며, 맵(Map)과 리듀스(Reduce) 두 단계로 구동된다. 따라서 하둡에서는 대용량 계산 작업을 수행할 때, 큰 파일을 블록 단위로 나누고 모든 블록에 대하여 동일한 Map 작업을 수행한 후 Reduce 작업을 수행한다.

우리의 이전 연구에서는 3D 프린터 출력을 위한 3D 모델을 G-code로 변환하기 위한 분산 알고리즘을 제안하였다[9]. 3D 프린터는 제품의 제일 아래쪽부터 위쪽방향으로 헤드를 이동하면서 잉크를 분출하여 출력을 진행한다. 따라서 하둡의 Master 노드는 3D 모델의 페이셋들을 일정 크기로 구분하여 N개의 Slave 노드들에게 전달하여 Z값에 따라 정렬하도록 한다. 이후 Master 노드는 이를 하나로 합친 후 N등분하여 Slave 노드들에게 전달하여 G-code로 변환하도록 하였다.

만약 3D 모델의 크기가 커지거나 대상의 정밀도가 높아진다면 제품의 표면을 구성하는 단위가 매우 작아지게 된다. 즉, 3D 모델의 크기가 커지거나 정밀도가 높아진다면, 모델을 구성하는 페이셋의 개수는 매우 많아지게 되며, 따라서 이로부터 G-code로 변환에 걸리는 처리 시간이 매우 커지게 된다. 이를 위해 본 논문에서는 하둡을 이용하여 이 변환작업을 수행하고자 한다. 3D 모델 파일에 대하여 일정한 전처리를 수행한 후, Map 단계에서 이 파일을 동일한 개수의 페이셋으로 구성된 블록으로 만들어 각 블록 내의 페이셋들을 최소 Z축 값에 따라 정렬을 수행한 후 Reduce 단계에서는 전체 블록을 하나의 정렬된 파일로 만든다. 앞서 언급하였듯이, G-code는 제품의 아래쪽(가장 낮은 Z값)부터 헤드의 이동 및 잉크 분출 명령을 수행하도록 하며, 이후 그 다음 높이만큼 이동한 후 계속 반복하게 된다.

이후 2장은 3D 프린팅 및 하둡과 관련된 연구들을 논의한다. 3장에서는 본 논문에서 제안하는 3D 모델을 G-code로의 변환 알고리즘에 대하여 설명하고, 4장에서는 하둡을 활용하여 제안한 알고리즘에 대한 변환 성능을 보여준다. 마지막으로 5장에서는 결론 및 향후 연구에 대하여 기술한다.

2. 관련 연구

본 연구과 관련된 주요 연구 주제는 3D 프린팅과 아파치 하둡이다. 따라서 두 분야로 구분하여 관련연구를 정리한다.

2.1 3D 프린팅

3D 프린팅이란 디지털 모델을 이용하여 각 레이어에 프린팅 재료를 뿌려 실제 개체를 제작하는 기술이며, 주로 적층제조(additive manufacturing) 방식을 의미한다. 이를 위해 먼저 CAD나 3차원 모델링 소프트웨어를 이용하여 3차원 모델을 생성해야 하며(Modeling), 이때 CAD와 표준 데이터 인터페이스는 주로 STL 파일 형식을 따른다. 생성된 3D 모델에 따라 3D 프린터는 물체를 제작하게 된다(Printing). 이 과정에서는 STL 파일을 읽어들여 프린터를 위한 명령어로 변환한 후 CAD 모델의 가상적인 단면에 프린팅 재료를 뿌려서 연속적인 층을 생성한다[4, 5].

3D 프린팅과 관련해서는 크게 프린팅 장비를 만드는 기술, 다양한 제품을 생성할 수 있도록 지원하는 3D 프린팅 기술, 그리고 3D 모델을 생성하는 기술로 구분할 수 있다. 하지만 장비나 재료 분야는 본 논문의 연구 주제와 상이하므로 더 이상 고려하지 않으며, 따라서 본 연구에서는 3D 모델 생성과 관련된 연구에 초점을 맞춘다.

앞서 언급하였듯이, 3D 프린팅을 하기 위해서는 3D 모델을 먼저 제작하여야 한다. 이와 관련된 연구들은 주로 제작하고자 하는 제품의 특성을 올바르게 반영한 모델 생성에 주력하고 있다.

[10]의 연구에서는 미세한 제품 표면에 대한 다양한 양방향 반사율 분포 함수를 처리할 수 있는 기법을 제안하였다. [11]의 저자들은 3D 프린팅 과정에서 힘-무게 비율에 대하여 관심을 가지고, 이를 해결할 수 있는 Surface Mesh 모델을 제안하였다. 이에 반해 [12]의 저자들은 실제 출력을 해보면 원래 예상했던 모델과 차이가 난다는 사실로부터 출발하여 Simulator를 제안하였다.

이러한 연구들은 목적에 부합하는 제품을 생성하기 위해 3D 모델 개발과 관련되어 있다.

2.2 아파치 하둡

아파치 하둡은 대용량의 구조화된 데이터뿐 아니라 비구조화된 데이터를 다루기에 적합한 도구이다. 하둡의 효율적인 MapReduce job scheduling 알고리즘으로 인하여 여러 장점 – 중복성(redundancy), 확장성, 병렬처리, 분산 구조 등 –을 얻을 수 있다. [13] 연구는 여러 MapReduce 알고리즘의 장점 및 단점, 주요 특징을 비교 분석하였다. 하둡의 주요한 특징 HDFS은 POSIX에 호환가능하지 않으며, 이에 따라 다양한 HPC에 적용하기 용이하지 않았다. 이에 [14] 연구는 이러한 문제를 해결하기 위해 하둡의 분산 파일 처리 특성을 활용하여 다양한 HPC 환경으로 적용가능한 MARIANE를 제안하였다. [15]의 저자는 하둡에서의 처리 비용과 시간 효율성에 초점을 맞추어, 퍼블릭 클라우드에서 동작하는 다목적 응용 프로그램의 최적화 기법을 제안하였다.

이처럼 하둡과 관련된 연구는 하둡 자체의 성능을 향상시키기 위한 알고리즘 개발이거나, 혹은 특정 응용 개발에 하둡을 활용한 사례에 해당한다.

3. Hadoop에서 3D 모델 데이터를 G-code로 변환

3D 프린터로 출력하기 위해서는 먼저 3D 모델을 생성한

Fig. 1. 3D Model Represented by Facets

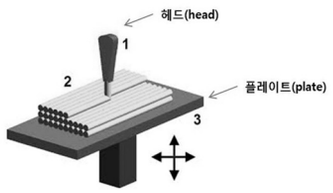


Fig. 2. 3D Printer

후, 이를 프린터가 인식할 수 있도록 G-code로 변환해야 한다. 본 장에서는 3D 모델의 특징에 대하여 간단히 설명한 후, 3D 모델 데이터를 G-code로 변환하는 알고리즘을 제안하며, 마지막으로 하둡을 이용하여 구현하는 방식까지 설명한다.

3.1 3D 모델과 G-code

일반적으로 3D 모델은 Fig. 1과 같이 3차원의 조그만 평면을 2차원 도형으로 표현한다. 그림에서는 삼각형으로 표면을 나타냈는데, 이러한 삼각형을 페이셋(Facet)이라고 부른다[8]. 하나의 페이셋에는 공간좌표 및 평면과 관련된 다양한 속성 정보를 담고 있지만, 본 논문에서는 좌표 정보만 고려한다. 3D 모델의 크기와 정밀도에 따라 차이가 있지만 하나의 모델을 표현하기 위해 매우 많은 수의 페이셋이 있어야 한다. 그리고 이러한 페이셋을 저장하고 있는 STL 파일에는 특별한 규칙없이 페이셋들이 저장되어 있다.

3D 프린터는 프린터의 종류에 따라 차이는 있지만, 대개 출력은 아래쪽(Z축으로 낮은 값)부터 위쪽 방향으로 진행된다. 일단 가장 낮은 위치로 플레이트가 고정되면, 헤드가 X-Y 축을 따라 이동하면서 적절한 위치에 잉크를 뿌리면 플레이트 위로 제품이 출력되게 된다. 현재 높이의 출력이 완료되면 플레이트를 일정 크기만큼 이동한 후 출력작업이 계속 진행된다(Fig. 2 참고).

이러한 프린터의 동작은 G-code에 의해 제어된다. G-code는 3D 모델의 아래쪽부터 위쪽 순서로 각 Z 위치에 해당하는 페이셋을 찾아서 명령어로 변환함을 의미한다. 이는 Z축 값과 무관하게 저장된 3D 모델의 페이셋으로부터 Z값에 영향을 받는 G-code를 생성해야 하며, 그 대상이 되는 페이셋의 개수가 많아진다면 계산 부하가 매우 커짐을 알 수 있다.

3.2 G-code 생성 알고리즘

3D 프린터는 Z축을 기준으로 가장 아래쪽을 출력하고, 일정 크기만큼 플레이트를 아래로 이동한 다음 다시 출력하는 동작을 반복한다. 따라서 G-code의 명령은 크게 '플레이트의 단위크기만큼 이동 명령', '헤드 이동 명령', '헤드 이동하면서 잉크 출력 명령'으로 구분할 수 있다. 이 중에서 헤드가 이동하면서 잉크를 출력하는 명령은 3D 모델에서 현재 플레이트의 높이(Z 축값)와 겹치는 페이셋을 찾아서 그 겹선에 해당하는 부분에 잉크를 뿌리는 것이다. 예를 들어, Fig. 3에서 현재 출력이 일어나는 부분(빨간색 부분)과 겹치는 페이셋을 찾은 후 그 페이셋에 대하여 잉크가 뿌려지는 부분을 계산해야 한다. 이러한 계산은 Fig. 4에서 보여준다. 각 페이셋은 삼각형이며, 각 꼭지점의 좌표를 가지고 있다. 예를 들어 Fig. 4의 경우, 점 P1, P2, 그리고 점 P3, P2로 만들어지는 직선의 방정식과 현재의 플레이트 높이(Z=α)의 교점의 좌표를 Equation (1)에 의해 계산할 수 있다.

$$\frac{x-x_2}{x_2-x_1}=\frac{y-y_2}{y_2-y_1}=\frac{z-z_2}{z_2-z_1} \quad (1)$$

이 수식에 의해 계산된 두 점 R1과 R2는 헤드가 이동하면서 잉크를 뿌리는 G-code로 변환되게 된다.

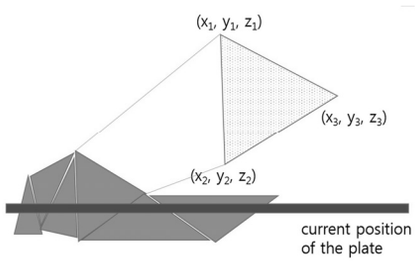


Fig. 3. The Parts Intersected with the Current Layer after Slicing

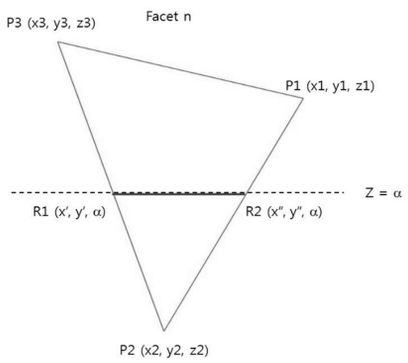


Fig. 4. Calculation of Intersected Line Between Each Facet and Current Layer(Red Line)

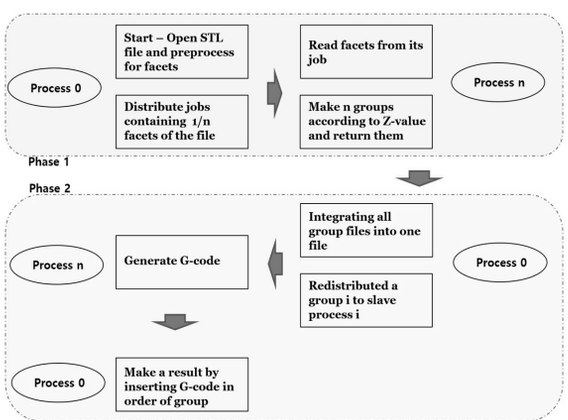


Fig. 5. Data Conversion Algorithm from Facets to G-code

Fig. 5는 페이셋으로부터 G-code를 분산 방식으로 생성하는 알고리즘을 보여주고 있다. 그림에서 보여지듯이, 프로세스 0은 먼저 페이셋들의 집합이 저장된 STL 파일을 T 등분하여 프로세스 1..T에게 전달하면, 그 프로세스들은 이를 각 페이셋의 가장 낮은 Z값에 따라 정렬하여 프로세스 0에게

돌려준다. T등분된 정렬데이터를 Merge sort 알고리즘에 의해 완전히 정렬한 다음, 정렬된 페이셋들을 모델의 높이에 T등분한 후 다시 프로세스 1..T 에게 전달하면 각 프로세스들은 받은 페이셋들을 이용하여 G-code를 생성할 수 있다. 이때 첫 번째 단계 뿐만 아니라 두 번째 단계에서 높이 h와 $h^{'}(h \neq h^{'})$에 대한 G-code 변환작업은 서로 독립적이므로 분산 작업이 이루어지더라도 문제가 없다. 왜냐하면 Fig 3, 4에서 보여지듯이, 하나의 페이셋이 프린터의 플레이트 위치 h, $h^{'}$에서 만나더라도 계산하는 G-code 생성은 수식 (1)에 의해 계산되며, 이는 다른 계산에 영향을 주지 않기 때문이다.

만약 3D 모델의 크기가 커지거나 정밀도가 높아진다면 표면을 구성하는 페이셋의 개수는 높이(h)의 세제곱(h^3)의 비율 혹은 정밀도(d)의 제곱의 비율(d^2)로 증가하게 되므로, 페이셋의 개수가 상당하게 증가하며, 그에 따라 상당한 변환시간을 소요하게 된다.

3.3 Hadoop을 이용한 변환 작업

작은 하둡 클러스터에는 하나의 마스터와 여러 워커 노드들로 구성 되어 있다. 마스터 노드들은 JobTracker, TaskTracker, NameNode, DataNode로 구성 된다. 슬레이브 또는 워커 노드는 DataNode와 TaskTracker로서 동작을 한다(Fig. 6. 참고). Fig. 5에서 프로세스 0은 마스터 노드의 역할이며, 워커노드는 프로세스 1..T에 해당한다.

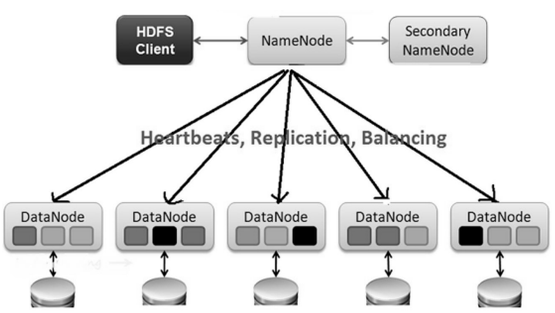


Fig. 6. Conversion using Hadoop

일반적으로 대용량 데이터 처리를 위하여 Hadoop을 활용하기 위해서는 입력 데이터에 대한 전처리 작업 - Map - Shuffling - Reduce 과정을 진행한다.

Fig. 5의 첫 번째 분산 처리 단계가 바로 전처리 작업에 해당한다. 이것은 저장된 페이셋들은 출력물의 높이(Z축 값)에 어떤 규칙성도 보이지 않지만, 이에 반해 3D 프린팅 작업이 대개 Z축을 기준으로 0부터 최대 높이까지 순차적으로 진행된다는 속성을 이용한 것이다.

이를 위해 1단계에서는 각 페이셋의 세 좌표값에서 제일 낮은 Z값(z_{min})과 제일 큰 Z값(z_{mzx})을 추출한 후, 이 값을 이용하여 정렬을 수행한다. 2단계에서는 출력 대상의 높이 값을 N등분한 후, 페이셋들의 z_{min}값에 해당하는 구간을 데이터 노드에게 전달하여 변환작업을 수행하도록 한다.

Hadoop은 모든 데이터 노드들이 HDFS에 저장된 데이터를 읽거나 쓰기 작업을 할 수 있다. Map의 결과로(key, value)쌍의 데이터가 생성되며, 이 데이터 집합은 Shuffling 과정에서 Key값으로 정렬된 후 HDFS에 저장된다. 이 데이터는 Reduce에 의해 읽혀지게 된다.

이 과정을 4단계로 구분하여 보다 구체적으로 설명한다.

1) 전처리

STL 파일에 저장된 페이셋은 3개의 공간 좌표값으로 구성되어 있다. 이 파일로부터 각 페이셋에 대한 공간 좌표값만 추출하여 문자열 형식으로 저장한다. 예를 들면, 파일로부터 하나의 페이셋의 좌표값(예를 들면, (10.3, 2.5, 8.3), (6.5, 4.9, 5.3), (7.5, 4.7, 10.6))을 읽어들인 후, 이를 문자열 “10.3 2.5 5.3 6.5 4.9 8.3 7.5 4.7 10.6”로 저장한다.

2) 데이터 노드에서 Map

이 단계에서 각 데이터 노드는 N등분된 전처리 파일을 읽어서 직접 G-code를 생성한다. 읽어들은 각 (key, value)에서 Value가 이전 전처리 단계에서 저장한 좌표값이다. Map이 이 페이셋을 읽으면 Fig. 7과 같은 과정에 따라 G-code를 생성한다.

예제로 든 페이셋은 높이(Z축값)가 5.3 - 10.6에 해당한다. 하지만 3D 프린터에서 한번에 출력하게 되는 잉크 자체가 일정 두께를 가진다. 그림에서는 그 두께가 0.5라고 가정한다. 따라서 그림의 윗부분처럼 0.5 단위마다 만나는 두 점의 좌표가 바로 프린터 헤드가 이동하면서 잉크를 뿌리는 위치가 된다. 그림에서 제일 아래 Z값은 5.3이지만, 플레이트의 위치는 5.0, 5.5, 6.0처럼 0.5씩 증가하게 된다. 따라서 Z값이 5.5 위치부터 교점의 좌표를 계산하여 그에 해당하는 G-code를 생성한다.

수식 (1)에서 z 변수에 현재의 플레이트 높이 값을 대입하면 교점의 좌표를 얻을 수 있다. 이렇게 얻어진 두 좌표 정보는 이후 Reduce를 위해 다시 (key, value)로 저장된다. 이때 Key는 높이이고, value는 교점의 좌표가 된다.

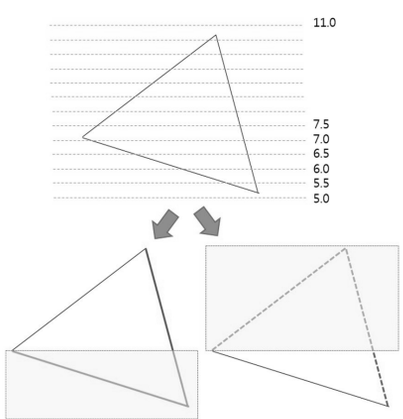


Fig. 7. G-code Generation from Each Facet

3) Shuffling

Map 단계는 하나의 페이셋으로부터 다수의 (key, value) 쌍을 생성하게 되며, 또한 다수의 페이셋이 변환 처리해야 한다.

Shuffling은 생성된 (key, value) 쌍을 key 값을 기준으로 정렬한다. 여기서 key는 출력이 이루어질 플레이트의 높이 값이라고 해석할 수 있다. 따라서 동일한 key값(즉, 동일한 높이)을 가지는 여러 value들이 묶여서 $(key, [value_1, value_2, ...value_n])$처럼 표현된다.

4) Reduce

이 단계는 Shuffling의 결과로 동일한 높이에 해당하는 G-code에 대한 최적화 작업을 수행한다. 현재 프린터의 플레이트 높이에서 헤드의 이동이 최소가 될 수 있도록 G-code 명령을 조절한다. 이는 key에 대한 value 값들의 위치를 변경하거나, 출력 위치와 위치 사이에 헤드를 이동하는 명령을 추가하는 것을 의미한다.

3.4 G-code 포맷

3D 프린터마다 사용하는 G-code는 다소 차이가 난다. 본 논문에서는 변환 소프트웨어로 널리 활용되고 있는 Cura를 이용하여 변환 작업을 수행한 후, 생성된 G-code를 이용하여 명령어를 역으로 추출하였다. 즉, 생성된 G-code는 앞부분의 환경 설정 부분 이후 다음과 같은 명령어들을 나열하는 방식을 취하고 있다. 여기서 G1 은 헤드의 이동을 의미한다.

G1 F1200 X77.453 Y75.701 E0.05826
G1 X79.332 Y74.083 E0.11656
G1 X81.327 Y72.611 E0.17486
. . .

4. 변환 결과

Hadoop은 5대의 컴퓨터 위에 설치되었으며(hadoop1 ~ hadoop5), 그 중 hadoop1이 마스터 노드이고 나머지는 워커 노드로 동작한다(Fig. 8 참고).

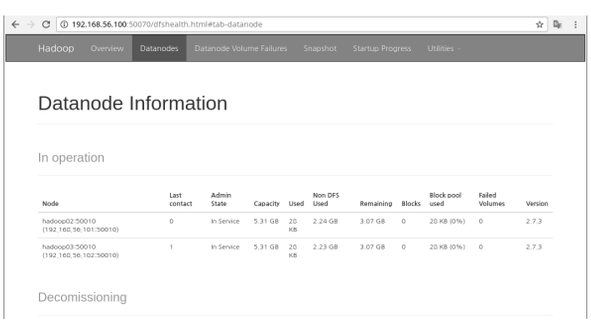
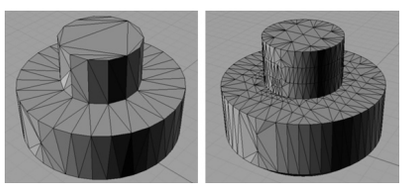


Fig. 8. Hadoop Cluster Setup

변환 작업을 수행하기 위하여 CAD를 이용하여 3차원 모델을 생성하였다. 생성된 모델에 대하여 높이값과 정밀도를 조정함으로써 포함된 페이셋의 개수를 변화시켰다. 가장 작은 크기(3cm)와 낮은 정밀도(=1.0)일 때 페이셋의 개수는 대략 3.6×10^4이었지만, 가장 큰 크기(15cm)와 높은 정밀도(=5.0)일때에는 페이셋의 개수가 1.8×10^7으로 약 500배 이상의 차이가 남을 알 수 있었다.

이처럼 변환 작업에서는 하나의 3D 모델 파일에 대해 정밀도와 크기를 변환함으로써 변환시간을 살펴보았다. 즉, 기본 3D 파일(정밀도 = 3.0, 객체 높이 = 9cm)에 대해 높이를 변화시키거나 정밀도를 변화시켰으며, 이는 모델 내에 포함된 페이셋의 개수를 변화시키게 된다.

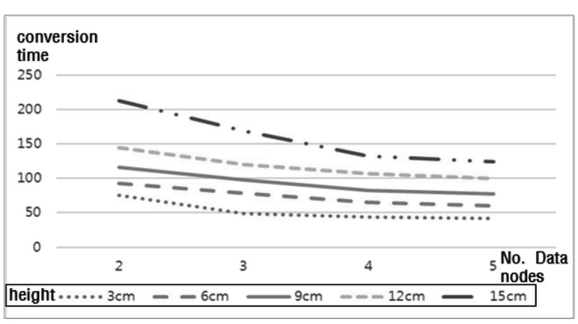


(a) Precision = 1.0 (b) Precision = 3.0

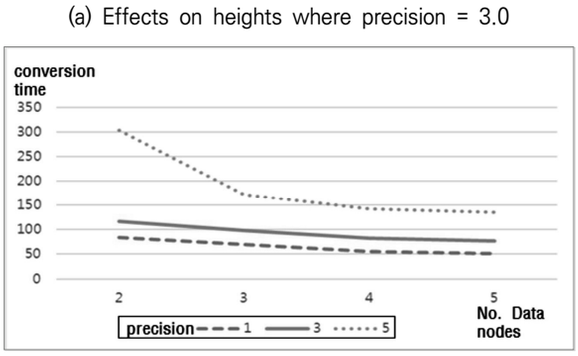
Fig. 9. Two 3D Models Represented with Different Precision

객체의 면은 페이셋의 집합으로 표현된다. 이때 정밀도는 페이셋의 크기에 영향을 받는다. 정밀도가 커진다는 것은 많은 수의 페이셋으로 면을 표현하게 됨으로써 객체를 보다 세밀하게 표현할 수 있지만, 페이셋의 개수가 매우 많아지게 되어 처리 시간이 커질 수 있다(Fig. 9). 3D 모델을 제작하는 소프트웨어에서 정밀도의 값을 지정하게 되는데, 그 값이 커지면 객체의 면이나 외곽선이 상대적으로 부드럽게 보여지게 된다.

Fig. 10에서 X축은 데이터 노드의 개수이고, Y축은 변환 시간을 의미한다. 첫 번째 실험에서는 정밀도를 고정하고 높이를 변화시키면서 변환 시간을 얻었다. 두 번째 실험에서는 모델의 높이를 9cm로 고정하고 정밀도에 따른 변환 시간 값을 나타내고 있다. 두 변환과정 모두 데이터 노드의 개수가 증가하면 분산도가 커져 변환시간이 향상되고 있으며, 또한 페이셋의 개수가 커지면 변환시간도 커짐을 알 수 있다.



(a) Effects on heights where precision = 3.0



(b) Effects on Precision Where Height = 9cm

Fig. 10. Conversion Time

두 실험에서 유의할 점은, 가장 변환시간이 오래 걸린 제일 위 경우이다. 예를 들면 Fig. 10-(a)에서 모델의 크기가 15cm로 커졌을 때 변환시간은 231초였지만, 나머지 경우는 140초 이하였다. 정밀도가 변화된 Fig. 10-(b)의 경우, 정밀도가 5인 경우에는 변환시간이 300초가 넘었지만, 나머지 경우는 100 내외였다. 본 실험에서는 비교적 단순한 3D 모델을 이용하여 변환 작업을 수행하였지만, 보다 복잡한 모델을 이용한다면 페이셋의 개수도 매우 많아질 것이다. 개발한 변환 프로그램에서 모델을 표현하는 페이셋의 개수가 5×10^5개 이상이 되면, 그 변환시간이 매우 커지게 됨을 알 수 있었다.

본 변환 실험에서는 단순히 높이만 변화시켰지만, 현실적으로는 높이가 증가하면 부피도 커져 페이셋의 개수는 더 많아질 것이며, 이 경우에는 변환시간의 단축 효과가 더 커질 것으로 판단된다.

5. 결 론

대용량 데이터 처리는 IT 분야의 중요한 연구 주제 중 하나이다. 3D 프린터를 이용하여 제품을 출력하기 위해서는 3D 모델을 먼저 생성한 후 이를 G-code로 변환해야 한다. 이때 모델의 크기나 정밀도에 따라 모델을 표현하는 페이셋의 개수가 매우 커지게 되며, 이러한 대용량의 페이셋을 처리하여 G-code를 생성하는 것은 큰 계산 부하를 발생시킨다. 이에 본 논문에서는 2단계 분산 알고리즘을 개발한 후, 이를 Hadoop 플랫폼의 Map-Reduce 기능을 활용하여 구현하였다. 그 결과 분산 작업 과정의 복잡도를 줄이면서도 효율적인 분산처리가 가능함을 알 수 있었다.

추후 더 복잡한 3D 모델과 더 많은 데이터 노드를 이용하여 대용량의 데이터 변환에도 적용하고자 하며, 이 작업이 원활하게 이루어질 경우 다양한 응용(빅데이터 처리나 기계학습을 위한 전처리 등)에도 적용할 수 있는 기반 기술로 활용하고자 한다.

References

[1] M. A. Beyer and D. Laney, "The importance of Big Data: A Definition," Gartner, 2012.

[2] D. Becker, T. D. King, and B. McMullen, "Big data, big data quality problem," *Int'l Conf. on Big Data*, pp.2644-2653, 2015.

[3] The Exponential Growth of Data, [Internet], https://insidebigdata.com/2017/02/16/the-exponential-growth-of-data, 2017.

[4] J. Teixeira, G. Barros, V. Teichrieb, and W. Correia, "3D Printing as a Means for Augmenting Existing Surfaces," *Symposium on Virtual and Augmented Reality (SVR)*, pp. 24-28, 2016.

[5] P. F. Jacobs, "Rapid prototyping & Manufacturing: Fundamentals of Stereolithography," Society of Manufacturing Engineers, 1992.

[6] G. Turkington, "Hadoop - Beginner's Guide," PACKT, 2013

[7] M. Bhandarkar, "MapReduce programming with apache hadoop," *Int'l Symp. on Parallel & Distributed Processing (IPDPS)*, Vol.1, pp.1, 2010.

[8] S. Vemula and C. Crick, "Hadoop image processing framework," *IEEE Int'l Congress on Big Data*, pp.506-513, 2015.

[9] K. E. Lee, M. J. Kim, D. Jeong, and S. Kim, "High volumes of data conversion based on Hadoop," *The KIPS Spring Conference*, C2-24, 2019.

[10] O. Rouiller, B. Bickel, J. Kautz, W. Matusik, and M. Alexa, "3D-Printing Spatially Varying BRDFs," *IEEE Computer Graphics and Applications*, Vol.33, pp.48-57, 2013.

[11] J. Wang, R. Zhao, and M. Pang, "Modeling Single-Gyroid Structures in Surface Mesh Models for 3D Printing," *Int'l Conf. on Cyberworlds*, Vol.1, pp.1-8, 2018.

[12] X. Luo, J. Wang, N. Liu, Z. Zhao, and Y. Zhou, "YaRep: A Personal 3D Printing Simulator," *Int'l Conf. on Virtual Reality and Visualization(ICVRV)*, pp.408-411, 2014.

[13] E. Mohamed and Z. Hong, "Hadoop-MapReduce Job Scheduling Algorithms Survey," *Int'l Conf. on Cloud Computing and Big Data*, pp.237-242, 2016.

[14] Z. Fadika, E. Dede, M. Govindaraju, and L. Ramakrishnan, "MARIANE: MApReduce Implementation Adapted for HPC Environments," *IEEE/ACM Int'l Workshop on Grid Computing*, pp.82-89, 2011.

[15] N. Alasmari, "Optimising Cloud-based Hadoop 2.x Applications," *IEEE/ACM Int'l Conf. on Utility and Cloud Computing Companion(UCC Companion)*, pp.43-46, 2018.

이 강 은

https://orcid.org/0000-0001-8416-0359
e-mail : gang3039@naver.com
2017년 ~ 현 재 서경대학교 소프트웨어학과 학사과정
2019년 ~ 현 재 서경대학교 DI연구실 학부연구원
관심분야 : 대용량 데이터 처리, 인공지능

김 성 석

https://orcid.org/0000-0002-7596-0757
e-mail : sskim03@skuniv.ac.kr
1997년 고려대학교 컴퓨터과학과(학사)
1999년 고려대학교 컴퓨터과학과(석사)
2003년 고려대학교 컴퓨터과학과(박사)
2001 ~ 2003년 한국산업기술대학교 컴퓨터공학과 겸임교수
2014년 University of Texas at El Paso 교환교수
2003년 ~ 현 재 서경대학교 소프트웨어학과 교수
관심분야 : 데이터베이스, 분산시스템, 대용량 데이터 처리, 기계학습