**Solid Freeform Fabrication 2016: Proceedings of the 27th Annual International
Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper**

Scalable Linking of Slice Layer Information with Process Monitoring Data in Additive Manufacturing Machines

[1*]A. Angrish, [1]S. Singh, [2]X. Shen, [1,3]Y.S. Lee, [1,3]P. Cohen, [1,3]B. Starly*
[1]Edward P. Fitts Department of Industrial and Systems Engineering, [2]Department of Computer Science
[3]Center for Additive Manufacturing and Logistics, North Carolina State University, Raleigh, NC 27695
*Contact authors: bstarly@ncsu.edu; aangrish@ncsu.edu

Abstract

In smart connected factories, manufacturing machines are capable of generating vast amounts of process data generated internally from within its control systems or from sensors coupled with the process. This streaming data must be stored and queried to perform data analytics or closed loop control to improve manufacturing processes. Currently, structured data schemas are ineffective in handling image and time-series data generated from additive manufacturing machines. In this paper, we propose an unstructured data schema through NoSQL document oriented database systems as an effective and scalable approach to capturing and storing real-time streaming data for process monitoring. In addition, we present an approach to linking in real-time, slice layer information and tag it with process related sensor data for performing fast, scalable queries either in real-time or post-fabrication. We have demonstrated our approach with two classes of additive manufacturing machines – Fused Deposition Modeling and Electron Beam Melting Systems from Makerbot and ARCAM respectively.

Keywords: Cybermanufacturing, Process Monitoring, NoSQL Databases, Networked Additive Manufacturing. MongoDB

Introduction and Motivation

Process monitoring of additive manufacturing processes, particularly those that produce high value added items, is a key critical step to ensure high quality parts. Given that there are a large number of parameters that can affect the final part geometry and properties, several approaches are taken to conduct process monitoring. Among the strategies are 1) Adding internal sensors that monitor the various motors and actuators of the machine; 2) Monitoring defects in a single layer, whether it be a powder bed, polymerized resin surface or a layer of extruded filaments; 3) Direct in-situ monitoring of the energy and material interaction zone. This zone can be the melt pool, filament extrusion and photopolymerization zone for each of the major classes of additive manufacturing machines. Any combination of these strategies will undoubtedly generate a lot of process data in raw numeric values or image based data. For example, a single build of the NIST additive manufacturing test artifact part on an ARCAM S10 generates above 300MB of raw numeric data without any additional sensors attached to the process. This time-series based data is stored as flat files on the local computer controlling the machine and accessible through machine specific proprietary software interfaces which allow users to analyze the log data generated during the build process. It is easily conceivable that managing the data generated from the process, specifically related to the storage, retrieval and cross-referencing of data among similar parts or multiple parts will be critical to downstream verification and validation exercises. Real-time streaming of process data also becomes critical for implementing close-loop control strategies to ensure a quality build.

With the networking of additive manufacturing machines and the addition of sensors to monitor the process becoming more prevalent, we argue that simply storing all of the data generated during the process in the form of flat log files or rudimentary databases will make the retrieval of this data limited to single part files. If further production scale analysis is desired, then custom software will have to be written to parse through multiple log files to extract needed information for statistical analysis or part validation exercises. More importantly, to assess the health of the additive machine, custom software will be required to analyze all of the process data locked into the multiple log files generated between scheduled maintenance interval times. The data to be analyzed will easily be in the multiple terabyte or more data block sizes, requiring production facilities to manage computing hardware to crunch and analyze through the data. In some instances, production scale AM machines archive the data or portions of it are deleted after a production run is completed. For example, on the latest generation of the ARCAM machines, x-ray images taken to interrogate layer build-up process is deleted after internal control systems determine there is nothing of value in the image.

With advancements made in sensor technology, particularly in metal based AM machines, we see a couple of challenges related to large data management and how this data is made useful for part verification or machine health status monitoring. In production scale settings, all of the process data, potentially with ‘Big Data’ type characteristics must be efficiently stored and retrieved. With advancements in distributed computing and storage, this challenge can be solved but must be architected in a manner suited for additive manufacturing machines. There are currently limited ways to relate sensor generated data and tag it to individual critical features on a product definition file. The potential benefit of tagging each feature on an AM part with associated sensor data generated during the fabrication of a particular part, allows a much more automated and robust verification/validation process.

This paper presents an approach on how the above challenges can be met. For the first challenge in efficiently storing time-series data, we present an approach to storing streaming sensor data in document based NoSQL database systems. This is in contrast to using relational type database systems such as those offered by MySQL, Oracle or Access, which are not suited to real-time streaming time-series based data. We have demonstrated this ability with simple Makerbot type additive fabrication machine. In another case study, we also show how log files generated during the fabrication of two different builds of the same part on an ARCAM machine is stored in a document based database system. To address the second challenge, we have shown an approach to relating the slice layer information associated with AM parts to sensor data generated during the fabrication of each layer of the part. Again this is shown for both FDM and Electron Beam Melting machines. At the end of the paper, we have identified challenges and future opportunities that can be pursued in relation to how data is stored, retrieved and cross-referenced in future cybermanufacturing type applications.

Related Background

Manufacturing intelligence driven by cross-connecting streaming data from physical machines can be a key driver to breaking down silos between manufacturing shop-floors, design teams, quality control and the several disciplinary teams that interface during a product lifecycle. As additive manufacturing machines are integrated into smarter manufacturing facilities and consequently with traditional manufacturing systems, it is critical that necessary software architectures are developed to interface with various hardware modalities on a shop floor.

a) Cybermanufacturing(CM) in Additive Manufacturing

The US National Science Foundation recent initiative in Cybermanufacturing illustrates why we need a data centric approach for additive manufacturing. The NSF defines - Cybermanufacturing "to be at the nexus of research advances from the engineering and computer/information science domains". One aspect of cybermanufacturing as illustrated by Lee et. al [1] and others is the role that streaming data can play in robust predictive and prescriptive analytics for machine state health monitoring. In another role of cybermanufacturing is how freeform designs made via ubiquitous design tools are easily transferred to conventional manufacturing or additive type machines. Another feature of CM is the enablement of technologies that make information systems to be fault-tolerant, scalable and enable the easy development of hardware agnostic software applications. To facilitate all of these CM features, there must be a robust mechanism for generation, transmission, storage, archival and retrieval of data. Additive manufacturing can play an important role in Cybermanufacturing due to the ability to readily transfer digital design files to print files with limited manual intervention. However, there is still more work to be done to allow AM machines to interface with other manufacturing machines or communicate with the 'digital thread' as promulgated by the digital manufacturing community.

b) Process Monitoring of Additive Manufacturing Machines

Process monitoring for AM machines is being actively researched to help improve part quality through closed loop feedback control. A common approach is through outfitting machines with external sensors and indirect offline measurements. For example, Dinwiddie et al [2], demonstrated the use of an extended range IR camera for quantification of temperature variation in 3D printed parts in FDM machines. Their aim was to understand how the variations in temperature affect the part strength. Faes et al [3] used a 2D laser triangulation for measurement of the thickness and the width of the extruded material. Li and Bian proposed [4] using a RGB camera for monitoring the AM process for printing hydrogels and studying the cross linking of the fibers. However, none of these methods store the data generated or propose any framework for feedback control of the AM machines. Work by Kim et al. [5] proposes "the development of a federated, information systems architecture for additive manufacturing". Their paper rightly points towards the need for a digital architecture for AM machines and needing access to the information generated at each point of the product development. Large companies like General Electric(GE) [6] are also working towards development of open architecture controllers for powder bed fusion machines which will enable third party developers to collect, analyze and use the data generated for monitor and control applications for these machines. Materialise [7] has a product called 'Streamics' which essentially streams build information from the AM machines to a centralized storage facility. With the software strength in process plan development, directly interfacing external sensor data to process layer information is yet to be shown.

Our proposed architecture differs from other methods in the fact that we are enabling development of third party applications on top of a NoSQL type database which allows for easy development of custom applications on this middleware type architecture. We use MongoDB as the choice for a NoSQL database due to its high scalability and ability to handle time series data efficiently. Another feature of our architecture is the ability to expand the data collection abilities from not just the machine, but also external sensors and databases.

c) Document based Database Systems for Storing Real-time Streaming Sensor Data

MongoDB is an example of a NoSQL database which allows for efficient storage and retrieval of time series data. Developed by MongoDB Inc., MongoDB uses a Javascript Object Notation (JSON) like document for data storage instead of a conventional table based approach used in SQL systems. Information within MongoDB [8] is stored in the following format (Figure 1). At the core of the MongoDB structure are documents, which are the most basic units of the database. Documents contain a JSON-like representation of the information stored in the database. Documents are analogous to rows in an SQL table. A bunch of documents form a collection. A collection is similar to a table in SQL, except no schema is enforced on the data stored within a collection. Above the collection exists the database system which is the container to hold single or multiple collections. The problem we face involves generation of vast amounts of unstructured machine data and sensor data, which needs to be collected, analyzed and stored. Since we are looking at scalability and ease of use, we decided to opt for the MongoDB. We collect the machine data and sensor data at a set polling frequency. The data collected is assigned a timestamp. At any given instance of time, a single or multiple data points may be stored. Therefore, the schema-less storage and the ability to create documents without a predefined structure of MongoDB is useful.

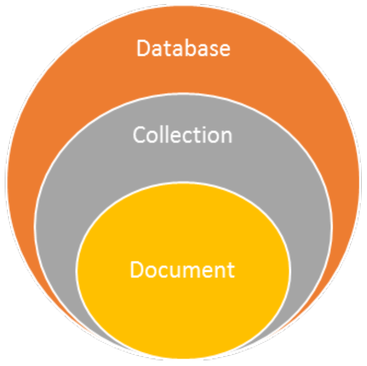


Figure 1 General Structure of MongoDB

Methods

<u>Information System Architecture for Networking of Additive Machines</u>

We designed our information system architecture for the purposes of networking additive manufacturing to have the three main properties:

1. Scalability: The architecture should be easily expandable and scaled easily across multiple machines.
2. Low cost: There are numerous solutions which are often tied together with the machine hardware. This increases cost in terms of creating necessary adaptors to allow physical machines to communicate with each other. Our architecture has to abstract out hardware level dependencies to enable ease of software development by third party.

3. Ease of development: We realized that for taking the full advantage of the information technology, the machines should be able to provide a platform for easy development of applications for monitoring and control for third-party developers.

One of the key outcomes which may be expected with the increasing interest in cybermanufacturing would be a future in which a manufacturing facility is transformed to one that involves a deeper integration of the digital realm with the physical world of machines, products, and manufacturing processes. This would in turn have the potential for improved quality control methodologies and reduction in costs associated with the same.

The purpose of our research is to include AM in the fast changing field of cybermanufacturing and develop an architecture for the same utilizing the APIs made available by the machine manufacturers and/or availability of open-source controllers in the machines themselves. The architecture is built on top of existing open-source technologies and is scalable to any number of machines. We are proposing a middleware architecture which would enable third party developers to do sophisticated monitoring and control applications without knowing the details of the lower level hardware, much similar to an Android [9] or an iOS for smartphones. Using this architecture we are proposing a method for associating product data (in the case of AM, the layer information) with process data from the AM machine and external sensors. The idea behind this is to demonstrate that with the availability of process information for each layer, we can enable monitoring, control and predictive analytics on AM machines.

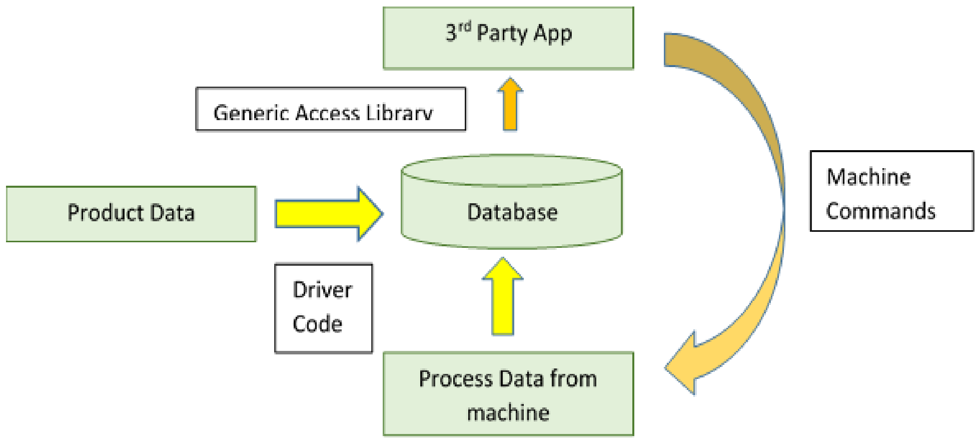


Figure 2: Overall High Level Architecture for Product and Process Data Association

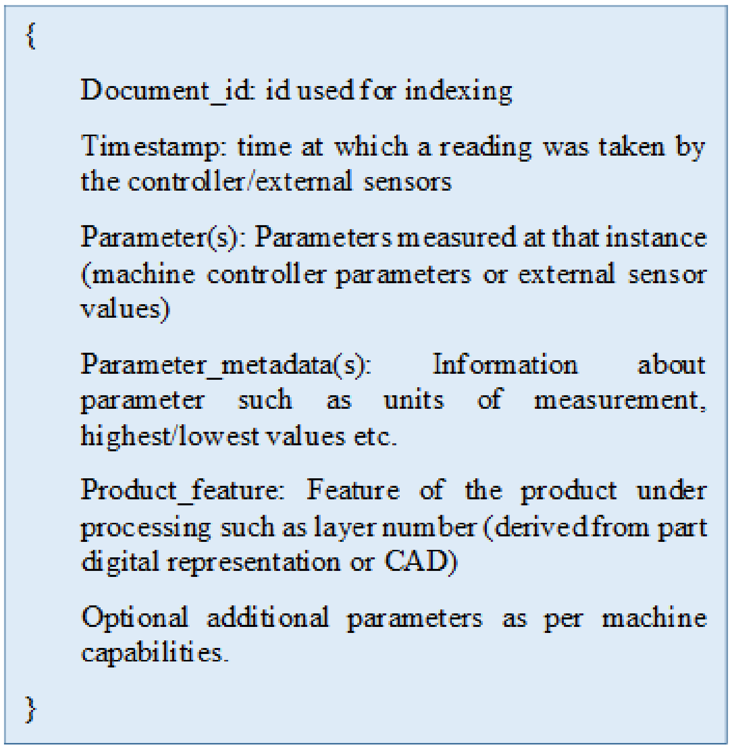
The inspiration for our architecture comes from the ubiquitous smartphones. In almost all smartphones, the end application developer does not need to know how the electronics inside the smartphones are connected, what pins are to be set high/low to trigger a particular action of the phone etc. All of this is standardized by the operating system (OS) of the phone. All the developer needs to do is make specific function calls responsible for a particular task and the embedded libraries within the OS take care of the rest. Similarly, our architecture provides an abstraction over the machine and allows storage, retrieval and manipulation of the enormous amounts of data generated by machines in an efficient manner along with control of machines, given the controller allows external commands in a programmatic fashion (see Figure 2).

The proposed architecture is composed of 3 parts: Driver, Database and a Generic Machine Access Library. Each of these components is described below:

1. The **Driver** is a piece of python code written specifically for each machine. The driver's purpose is to access the machine controller and any attached external sensors to the machine, combine that information with the digital/process file of the part being manufactured and push this information into a database. The driver codes will have functions to collect appropriate sensor data, identify the product feature under processing within the machine, collect sensor state and push all of it into a user defined object. The driver is capable of doing this repeatedly at a set polling frequency as per requirement of the end user. This object is then pushed into the database.

2. The **Database** implemented in the proposed architecture is a NoSQL type database. NoSQL databases give a high level of ease and are ideally suited for storing time-series based data. We have implemented a MongoDB NoSQL system since it provides features such as auto-sharding, easy scalability and high frequency of reading and writing data [10]. MongoDB is completely open-source and an easy to setup database system with numerous APIs for accessing the database in multiple languages such as C, C++, Python, Java etc. Therefore, the drivers may be written in any of the most popular languages and MongoDB will still be able to ingest and serve data as required. The database is further divided into 3 layers:

   a. A Raw Data Layer: As the name suggests, this layer contains all the raw data that is generated during the manufacturing of a part in the machine. The drivers continuously collect information from the machine and store it in the form of documents. Each document in the raw data layer follows a specific schema as shown in Figure 3:



{

Document_id: id used for indexing

Timestamp: time at which a reading was taken by the controller/external sensors

Parameter(s): Parameters measured at that instance (machine controller parameters or external sensor values)

Parameter_metadata(s): Information about parameter such as units of measurement, highest/lowest values etc.

Product_feature: Feature of the product under processing such as layer number (derived from part digital representation or CAD)

Optional additional parameters as per machine capabilities.

}

Figure 3: Document Structure Inside Each Collection

   b. An Information Layer: The information layer contains relevant information about the machine from which the data is being pulled by the drivers. This layer is populated only once when the first time a machine in installed. The information layer contains information about the machine name, version number, technology used, power consumption ratings, materials used etc.

c. Summary Layer: The summary layer contains a summary of all the parameters captured by the raw layer, relevant statistics pertaining to them (mean, mode, kurtosis etc.) and parts created using the machine.

These three layers are accessible to the application developer. The developer can access these layers using the generic machine access library as described below.

3. The **Generic Machine Access Library**: We have built a generic access library in Python which allows end users (in our case developers) to build their own applications for the machine. The library takes care of several aspects of data pulling and command issuance so that the user does not have to deal with machine specific commands. The library also offers an object oriented approach to programming. As a result, the application developers simply have to initialize an object referring to the machine and they can directly monitor and control the machines. It offers several functions such as getMachineData(layer) [for pulling specific layer data] or sendCommand(command) [for sending commands to the machine]. The generic access libraries give us the ability to analyze and retrieve information intelligently. It allows us to connect to the databases and pull relevant data, along with some basic in-built algorithms. The system can also utilize external libraries or user defined functions. As such, the developer needs to know only one language (Python, for example) and they can develop programs for any hardware without knowing internal details for the hardware. Together this structure can be shown diagrammatically in Figure 4.

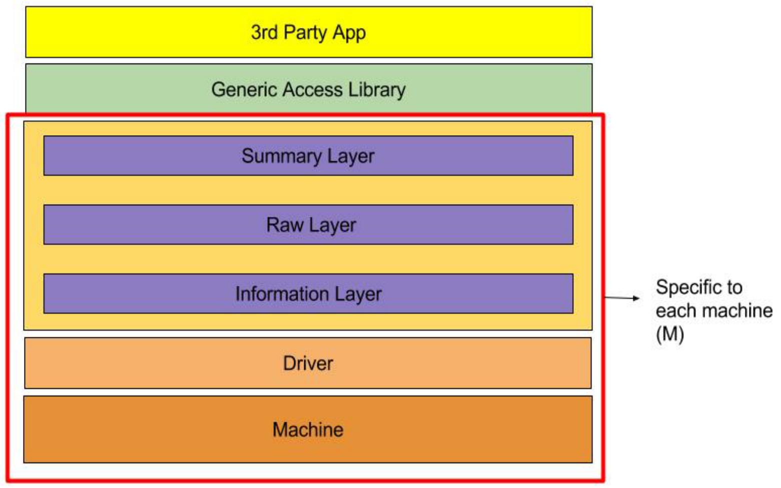


Figure 4: Overall Architecture for Third Party App Ecosystem development

The machine's driver and the database are specific to a machine. Therefore, if we have N machines on a shop-floor, we will have N such frameworks. However, the library on top of them can be used for all the machines. Therefore, a single app can be used for accessing and analyzing the process information from multiple machines We can also have multiple machines being monitored/controlled using a generic architecture as shown in Figure 5. Multiple machines will be reporting its data to the database associated

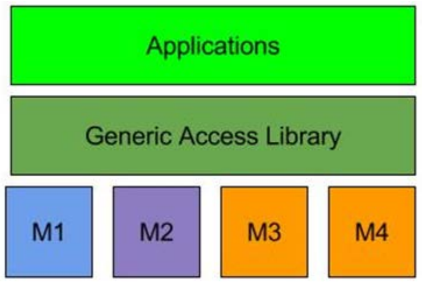


Figure 5: Architecture for multiple machines

with each machine. It is conceivable that a database of databases can be designed to direct the control of client requests to multiple machines.

Results

Application in FDM and EBM machine:

We chose 2 very different AM machines to demonstrate a proof of concept of our architecture. The first machine we chose was a Makerbot FDM machine and the 2nd is an ARCAM EBM (Electron Beam Melting) Machine. The Makerbot has a relatively open architecture compared to the EBM machine. We were able to get access to the Makerbot controller via the s3g protocol [11] (a custom communication protocol built on top of pySerial by Makerbot for communicating with Makerbots programmatically). Python based drivers for the Makerbot collects the following information from the Makerbot controller: *Extruder(s) in use, extruder temperature, extruder status, platform temperature, motherboard status, axis positions, file being printed, time stamp and layer number.* The driver collects this information and pushes it into a dictionary object in Python. This dictionary is then sent to MongoDB.

The EBM machine is a closed architecture system with no direct access to the controller. However, the machine generates very descriptive machine logs during the print operation. Scripts were written to read these log files, parses them and generates a dictionary. The ARCAM machine can keep a track of more than 15000 parameters for a single build. The number of parameters which actually vary during the build fluctuate from part to part. We examined several parts where the number of parameters vary from 700-900. Amongst the various parameters collected by the machine, some of the significant ones collected are as follows: *Build height, layer number, temperatures, timestamp, alarm status, current, power consumption etc.* This data is pushed into the raw layer of the machine as per their incidence in the log files. Once a build is completed, the machines summary and information layers are also updated automatically by the driver.

Case Study 1: With Makerbot FDM

This particular app written for the Markerbot is designed to capture machine axis movements and rebuild the part digitally by collecting data from the controller via driver codes, the part file and any associated sensor data. We will be using this raw data to identify the parts that parts were printed on the Makerbot, associate and visualize sensor readings with the slices of the part. This can be especially useful when there are fleets of FDM printers and the need for traceability on part validation and verification, particularly when process monitoring data is

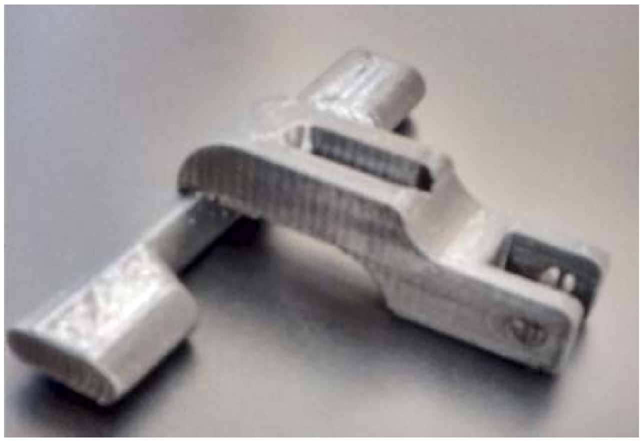
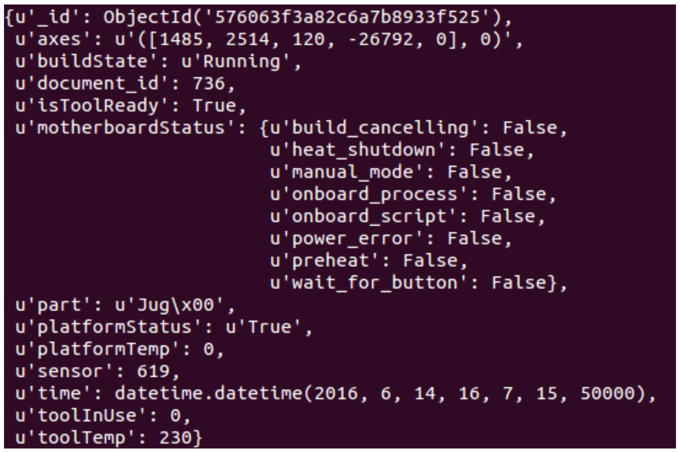


Figure 6: Actual part made on Makerbot

gathered. Moreover, by associating process information with product information, it becomes possible to identify differences at a slice level between products.

Here we have a part that was printed under a 0.1 mm resolution on a Makerbot printer. The machine was enabled to transfer the movement, temperature and sensor information to a database. We wrote a Python app to pull this information and plot the machine movement in a 3D scatter plot using the generic access library and the matplotlib library in Python (for plotting). For clarity, we have reduced the point size of the scatter plot. A sample document stored in the database is shown in Figure 7. As seen, external application can easily retrieve file names, information about the motherboard status, build state, platform and tool temperatures along with the associated sensor values, all synchronized by a timestamp. This information is collected 10 times every second as the printing process runs. Using the app, we can see in Figure 8, the part was built with a hexagonal infill. The side view and the iso views of the part are shown in Figure 8:



```
{u'_id': ObjectId('576063f3a82c6a7b8933f525'),
 u'axes': u'([1485, 2514, 120, -26792, 0], 0)',
 u'buildState': u'Running',
 u'document_id': 736,
 u'isToolReady': True,
 u'motherboardStatus': {u'build_cancelling': False,
                        u'heat_shutdown': False,
                        u'manual_mode': False,
                        u'onboard_process': False,
                        u'onboard_script': False,
                        u'power_error': False,
                        u'preheat': False,
                        u'wait_for_button': False},
 u'part': u'Jug\x00',
 u'platformStatus': u'True',
 u'platformTemp': 0,
 u'sensor': 619,
 u'time': datetime.datetime(2016, 6, 14, 16, 7, 15, 50000),
 u'toolInUse': 0,
 u'toolTemp': 230}
```

Figure 7: Sample document in MongoDB

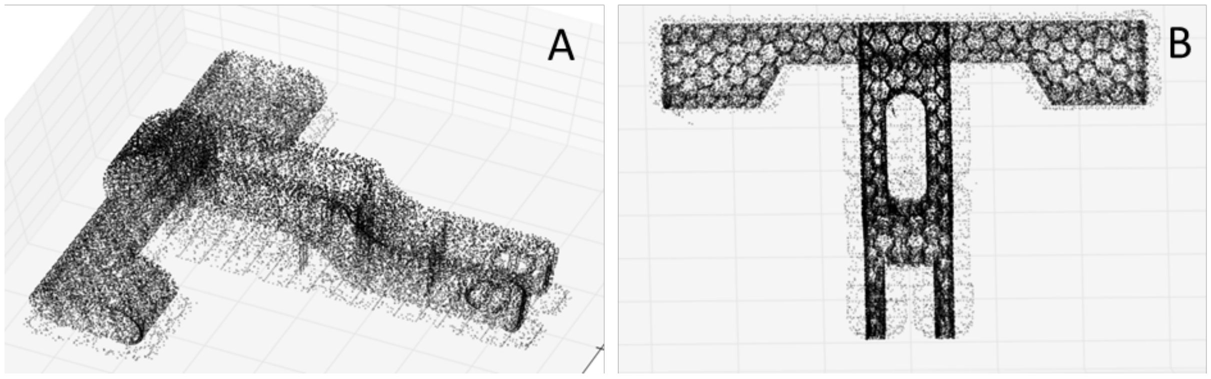


Figure 8: ISO and TOP Views of Digitally Reconstructed Parts from Machine Axis Data

The summary layer of the machine contains statistics for various parts including the mean, minimum and maximum values for the sensors. We can plot all the layers which have a sensor value more than the average of the values in the summary layer. We see the following plot, the sensor readings were more than the average for around half the part (slices exceeding sensor reading averages are represented in red and the part in green).

Such an analysis can be used for enhanced study of FDM printed parts and may also be used as a "playback" system for the machine. With the attachment of sensors such as rotary encoders, it will be possible to identify the actual movement of the motors vs the commanded motor movements for the FDM printer. By storing the values of the encoders, it will be possible to make an app which can compare the commanded movements with the actual movement of the axes. Such a system can be used for complete quality control of 3D printed products for every single part produced. It can also be useful when security implications are considered. Advanced pattern recognition apps can be written to identify deviations of part fabrication from the norm to warn users of errors and potential malfunction.

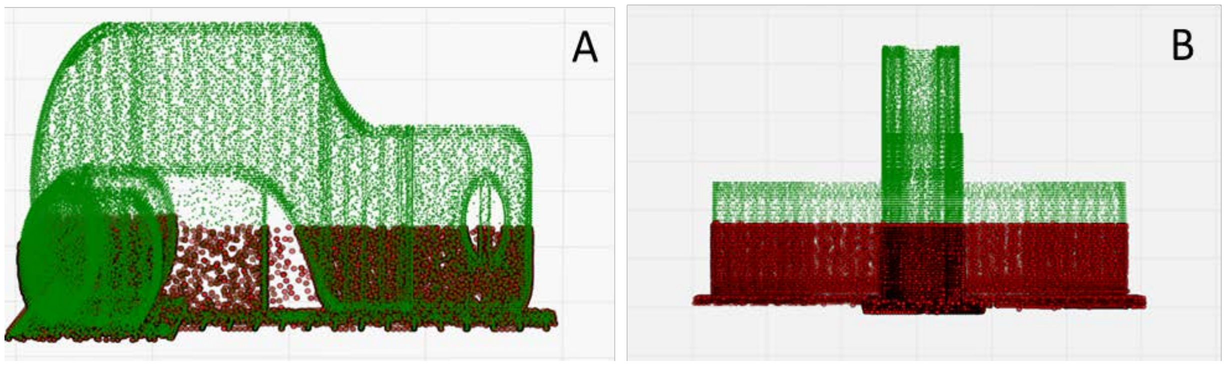


Figure 9: Side(A) and Front Views(B) of Reconstructed Parts with Sensor Values Overlaid (B) shows retrieval of reconstructed layers that have exceeded a certain value. This allow enabling qualification and verification processes.

Case Study 2: With ARCAM EBM

The Electron Beam Melting machines are far more complex and are a topic of intense research both by statisticians and material scientists. We demonstrate the use of our architecture for development of an app which can be used to facilitate research into development of analytical models and data visualization. In this app, we demonstrate the use of the generic access libraries along with the matplotlib and numpy packages in Python to develop regression models and data visualization tools for the ARCAM machines. This is unique such that it allows researchers to go above and beyond the basic visualization tools offered by ARCAM in their Log Studio software.

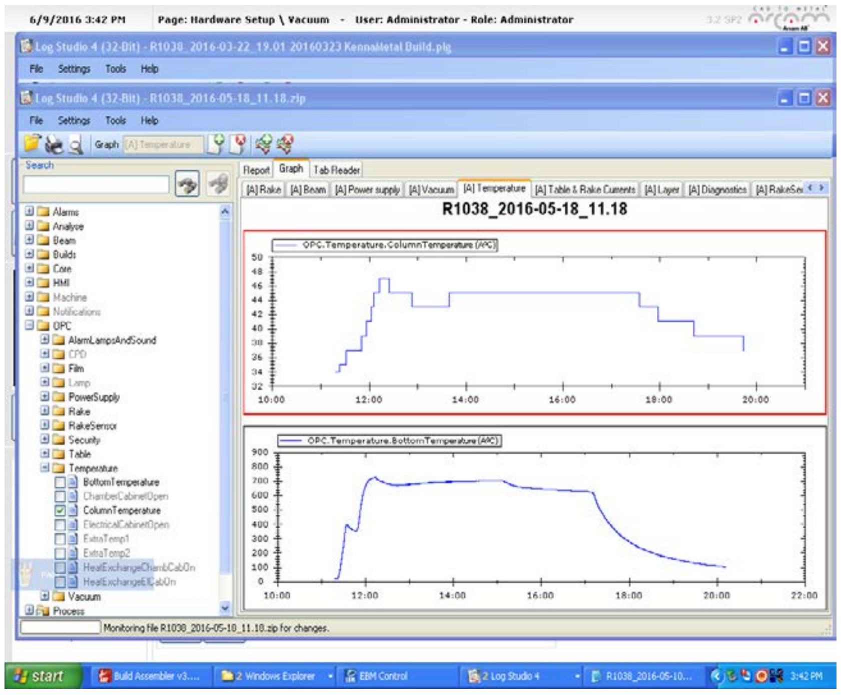
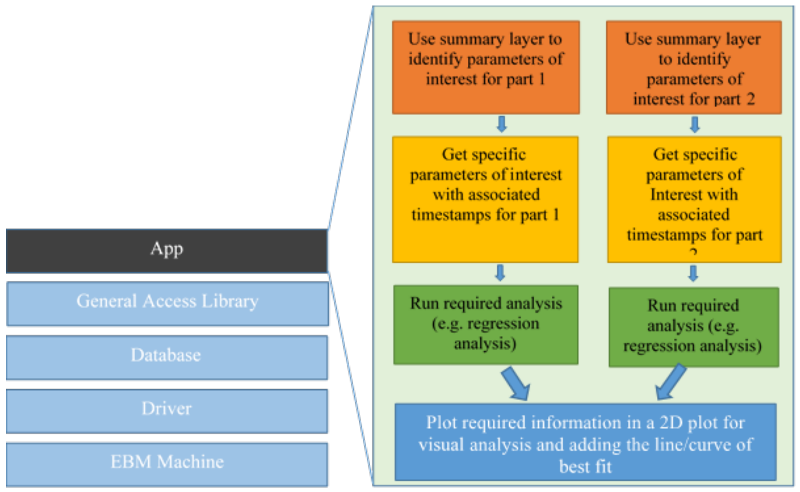

6/9/2016 3:42 PM
Page: Hardware Setup \ Vacuum - User: Administrator - Role: Administrator
Log Studio 4 (32-Bit) - R1038_2016-03-22_19.01 20160323 KennaMetal Build.plg
Log Studio 4 (32-Bit) - R1038_2016-05-18_11.18.zip
File Settings Tools Help
Graph
Report Graph Tab Reader
R1038_2016-05-18_11.18
OPC.Temperature.ColumnTemperature (APC)
OPC.Temperature.BottomTemperature (APC)
Alarms
Analyze
Beam
Builds
Core
HMI
Machine
Notifications
OPC
Temperature
BottomTemperature
ColumnTemperature
Vacuum
Process
Monitoring file R1038_2016-05-18_11.18.zip for changes.
start

Figure 10: Sample Log Studio Interface from ARCAM Software

Log Studio Software package is developed by ARCAM to the customers of their machines for analysis of the log files of the parts being built using the EBM machines. The software is meant for analyzing the build statistics for a particular build in the machine. A major limitation of the Log Studio is the inability to analyze multiple parts at the same time. As such, it becomes very difficult to compare multiple parts and multiple process features simultaneously without writing excessive code to compare data across multiple log files. We will demonstrate a simple app using our architecture to compare more than one part at a given time.

The app's algorithm framework is shown in Figure 11. We built two apps for the EBM. One is an app which requires a user to select a parameter and the app fits a curve to the data for modeling. As an example, we modeled the time for melting of a particular layer as the build proceeds for the fabrication of the NCSU football. We demonstrate our result in the following graph (generated via the app) in Figure 13. As can be seen, the time taken for melting the powder for each layer (contour) is low at the beginning and at the end of the build. But it increases as we move to the middle of the part. Just by interfacing with the log data generated, it is possible to identify the general contour shape of the part.


App
General Access Library
Database
Driver
EBM Machine
Use summary layer to identify parameters of interest for part 1
Use summary layer to identify parameters of interest for part 2
Get specific parameters of interest with associated timestamps for part 1
Get specific parameters of Interest with associated timestamps for part 2
Run required analysis (e.g. regression analysis)
Run required analysis (e.g. regression analysis)
Plot required information in a 2D plot for visual analysis and adding the line/curve of best fit
Figure 11 App Algorithm Framework

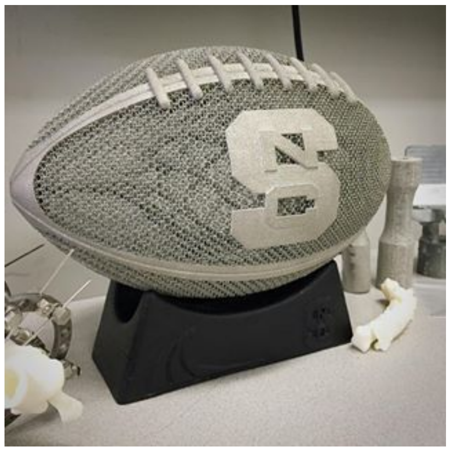


Figure 12: Porous Mesh Football built on the ARCAM EBM machine

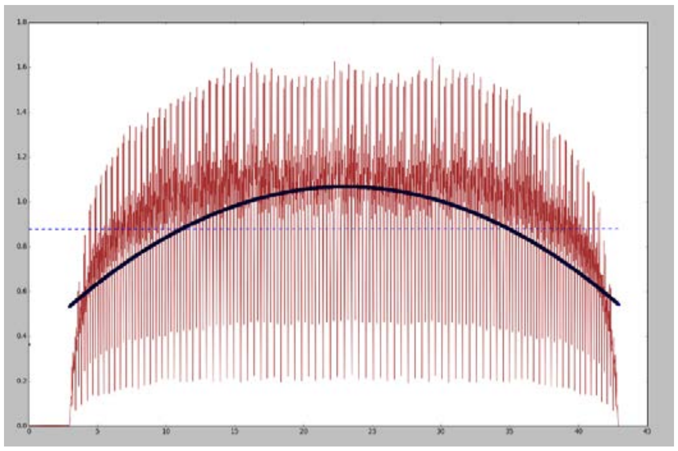


Figure 13: Contour Time vs Build Time. Due to the shape of the football, it is expected that melting time to create a contour is high in the center.

Two such footballs were printed with varying properties. Since the machine generates a huge amount of data (more than 900 parameters are recorded and measured during the build process), we decided to demonstrate the use of our method to compare parameters for both footballs that was only different in its orientation during the build process. Regression model for the curve: Time for a contour at any given point of time since the start of the build

$$y= t^2 * (-0.0013) + t*(0.0608) + 0.3679$$

where t is the time elapsed since the build started.

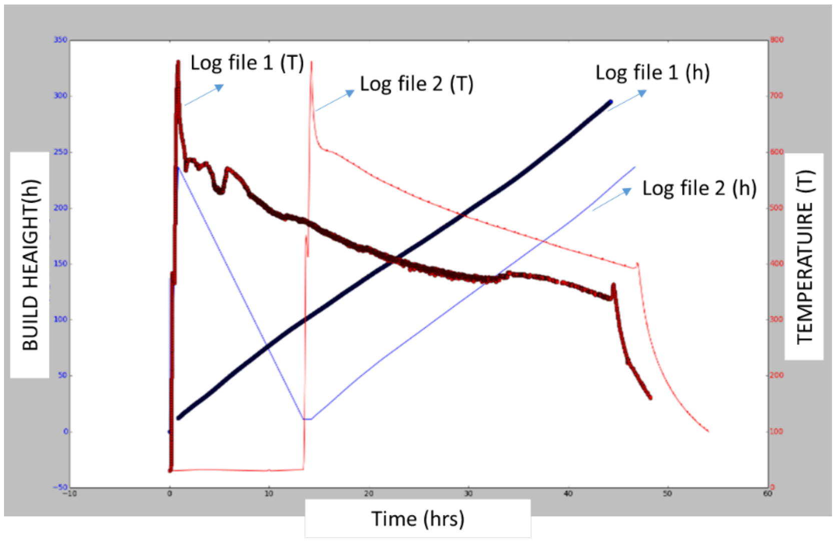


Figure 14: Build Height and Temperatures plotted against time for 2 machine log files created from the same model part.

The second client app reads the database for both the parts and plots a movement of the part table in the Z direction (which can be correlated with the height of the part), the temperature measurement at the bottom of the table and the time of the build (i.e. when the time data recording was started). Figure 14 represents the data we have plotted in a single chart for comparison of two part builds (same football model) with different orientations in the EBM machine. The bold dark marker (file 1) represents version 1 of the football and the thin line marker represents version 2 of the football. We notice that while the bottom temperatures (indicated using red color) of the two builds are similar, they are not identical. We notice there are differences after the peak in both plots. An interesting feature we can notice from the data is that for the football version 2, the build time is 46 hours but the height of the table goes down at first and then increases. This is explained using the bottom temperatures. The temperature is constant for the first 15 hours till which point the height reduces. This indicates that machine table position was simply moving with no actual printing taking place. This is not the case with the version 1 of the football. While we cannot ascertain the reason for this difference, it is apparent now that through the app, automated algorithms can be built to crunch through all of the log files to find anomalies. This can assist in process monitoring and qualification. Another point we can readily note is the difference in the height of the two versions. This difference can be due to several reasons such as orientation of footballs during the printing process. The importance to note is that such comparisons can be scaled to any number of parts. Models surrounding the data streamed from the electron beam melting machines enables better data analytics. The middleware architecture also enables any third party to build apps to extend the functionality of the machines.

Discussion

As we have shown in both the case studies, our library and architecture provides a layer of abstraction for both 3rd party developers and researchers which can be used for academic or industry related development. We have demonstrated the use of our architecture for development of data analytics tools for additive manufacturing machines. The results can be graphical as well as analytical. The project benefits from the use of a NoSQL database which allows storage of information in a schema-less format, freeing the driver developer to focus on collection of information and not in devising the schema. Also the inherent advantages of storing information in a document based structure since we store incoming data in the form of a time series data structure. As a result, it becomes possible for us to associate every reading with a time stamp. The results we have seen are dependent on the requirement of the application developer. Since the data is readily available, the app developer only needs to import the generic access library and use existing Python frameworks for making their task of app development faster.

There are numerous challenges and improvements to this work. The generic access library needs to be expanded to incorporate advanced numerical analysis. The graphing functions are currently implemented at the application level, which requires the app developer to have significant understanding of the libraries they are using (e.g. matplotlib and numpy). This can be incorporated in the library to unburden the developers from worrying about the graphs.

It is also possible for us to develop the generic access library for popular languages such as C, C++, Java, R etc. to suit the need of different academicians/industry. In fact this might be a necessity as Python is not the fastest of the languages in terms of execution. One major challenge here is that the libraries depend on the public availability of drivers for the language to interact with MongoDB. While they are available for most popular languages, some of the more exotic/proprietary languages will lack such drivers. As such, there will be a need to develop suitable workarounds.

More work needs to be done on better understanding the data from the additive manufacturing machines for identification of parameters which affect the printing process. For example, there are over 15000 parameters which can be tracked by the EBM machine. We are only able to track a few. Since we have all the information generated by the machine, it should be possible to implement advanced statistical and/or deep learning techniques to predict properties of the parts which are being printed in the EBM machine.

Another area of improvement could be development of a big data framework on top of MongoDB in order to speed up queries. While MongoDB is quite stable and scalable for large datasets, reduction of query times will result in faster code runs and might enable us to have real time control of AM machines based on sophisticated deep-learning algorithms running in real time.

Conclusion

We demonstrated the use of a completely open-source system for real-time monitoring and storage of manufacturing data for additive manufacturing machines. The system was built using custom driver software for each AM machine using Python which stores the data generated along with the slice information in a NoSQL database providing a scalable and easy way for associating production data with design information. We also developed a generic access library which provides APIs for extraction and analysis of this data from the database along with option control commands where available (in this case, Makerbot printers). We demonstrated the use of our architecture for 2 machines, Makerbot and ARCAM EBM machine. For the Makerbot case study, we demonstrated the ability to use the data generated for replaying what was developed in the machine and identifying the sensor reading values at different layers in the part manufactured. The EBM data was used to demonstrate the ability to perform regression analysis and to perform simple visual analytics on the build process.

Acknowledgement

We would like to thank Dr. Tim Horn and Austin Isaacs for providing their valuable help with the EBM machines. We would also like to thank the ISE department and the NSF for providing the resources for this research.

References

[1] Lee, J., Bagheri, B., & Jin, C. (2016). Introduction to cyber manufacturing. Manufacturing Letters, 8, 11-15. doi:10.1016/j.mfglet.2016.05.002

[2] Dinwiddie, R. B., Love, L. J., & Rowe, J. C. (2013). Real-time process monitoring and temperature mapping of a 3D polymer printing process. Thermosense: Thermal Infrared Applications XXXV. doi:10.1117/12.1518454

[3] Faes, M., Abbeloos, W., Vogeler, F., Valkenaers, H., Coppens, K., & Ferraris, E. (2014, September). Process monitoring of extrusion based 3D printing via laser scanning. In PMI 2014 Conference Proceedings (Vol. 6, pp. 363-367).

[4] Li, Y., & Bian, P. Y. (2014). A Strategy for On-Line Monitoring the Crosslinking Degree of 3D Printing Hydrogel Fiber Based on Dual-Threshold Enhancement Method. In Key Engineering Materials (Vol. 621, pp. 38-43). Trans Tech Publications.

[5] Kim, D. B., Witherell, P., Lipman, R., & Feng, S. C. (2015). Streamlining the additive manufacturing digital spectrum: A systems approach. Additive manufacturing, 5, 20-30.

[6] A Flexible Adaptive Open Architecture to Enable a Robust Third-Party Ecosystem for Metal Powder Bed Fusion AM Systems (4051). (n.d.). Retrieved July 14, 2016, from https://goo.gl/Q3H4rP

[7] Materialise Streamics. (n.d.). Retrieved July 14, 2016, from http://goo.gl/SclisW

[8] Glossary. (n.d.). Retrieved July 14, 2016, from https://goo.gl/7CCyB2

[9] Andrus, J., & Nieh, J. (2012). Teaching operating systems using android. In Proceedings of the 43rd ACM technical symposium on Computer Science Education (pp. 613-618). ACM.

[10] Kalan, M. (2015). MongoDB in Chicago: Benefits of Using MongoDB Over RDBMS. Retrieved July 14, 2016, from https://www.mongodb.com/presentations/mongodb-chicago-benefits-using-mongodb-over-rdbms

[11] Makerbot/s3g. (2013). Retrieved July 14, 2016, from https://github.com/Makerbot/s3g