Remotely Controlled Manufacturing: A New Frontier for Systems Research

Harsha V. Madhyastha
University of Michigan
harshavm@umich.edu

Chinedum Okwudire
University of Michigan
okwudire@umich.edu

ABSTRACT

It has now become commonplace for applications that run on smartphones, IoT devices, and even personal computers to rely on services hosted either in the cloud or on edge servers. Some of the motivations for this trend—augmenting thin clients and enabling a shared view across users/devices—also apply to manufacturing machines such as 3D printers, laser cutters, and machine tools. Off-device control of manufacturing devices can help outdated machines benefit from the most advanced control algorithms, which they are incapable of running locally. Cloud/edge support can also enable coordinated manufacturing of a product's parts across multiple machines, thereby reducing production time and cost compared to the use of a single machine. In this paper, we highlight the challenges in realizing these benefits for the manufacturing domain.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**; **Reliability**.

KEYWORDS

manufacturing, cloud computing, edge computing

ACM Reference Format:
Harsha V. Madhyastha and Chinedum Okwudire. 2020. Remotely Controlled Manufacturing: A New Frontier for Systems Research. In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications (HotMobile '20), March 3–4, 2020, Austin, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3376897.3377862

1 INTRODUCTION

Over the last 10–15 years, we have witnessed a dramatic shift in how software is developed and deployed. Instead of simply shipping binaries that we run completely locally on our devices, a wide variety of application providers now also rely on services hosted either in the cloud or at the edge. Examples include file storage (e.g., Dropbox), document editing (e.g., Google Docs), augmented reality (e.g., Microsoft Hololens), and gaming (e.g., Pokemon Go).

While there are a range of motivating factors that underlie use of the cloud/edge in software applications, two considerations are most relevant for our discussion here:

- To circumvent the limitations of client devices with respect to compute, memory, network, fault-tolerance, etc.
- To enable a consistent view of application state across devices and users

Most of the work in enabling this paradigm shift has, however, focused on applications that run on electronic devices such as laptops, smartphones, smart glasses, etc. In this paper, we shine light on a new setting which can benefit from cloud/edge support for the same reasons mentioned above: *control of manufacturing devices* (or machines) such as machine tools, 3D printers, and laser cutters.

First, in contrast to electronic devices, which users upgrade every few years, manufacturing devices are typically replaced significantly less frequently, e.g., every ten years [7, 11, 20]. As a result, they are often incapable of running the latest algorithms available to control their execution [10, 15, 25]. For example, there is a tradeoff between the speed with which a manufacturing machine is operated and the quality of the resulting product [1]. Advanced control algorithms help reduce the time needed to execute a job without adverse effects on the quality of the output product, e.g., by adaptively varying the speed of execution based on predictions of the machine's vibrations [8]. However, devices which are not state-of-the-art lack sufficient compute and memory resources to run such advanced controllers [17].

Second, to complete the production of any particular item, a manufacturing device can often take several hours. Consequently, when one wants to manufacture either a product composed of many parts or several copies of the same product, faster and lower-cost completion can be achieved by distributing production across multiple machines. Partly as a result of such motivations, a "manufacturing as a service" market is emerging, in which many users are making their manufacturing devices available for use by others [2, 13].

Remotely controlling a manufacturing machine's execution can therefore aid manufacturing by 1) enabling the use of computationally heavyweight control algorithms which the machine is incapable of running locally, and 2) offering a unified view across several machines. In addition, moving the execution of a machine's controller off the device also offers other benefits such as 3) enabling a faster update cycle of the controller software, and 4) helping the controller driving one device learn and adapt its execution based on observations made when previously operating other devices of the same type.

However, to realize these benefits, one cannot simply forklift solutions from traditional settings where consumer software relies on remotely hosted services. In particular, two characteristics primarily differentiate manufacturing machines from electronic devices.

HotMobile '20, March 3–4, 2020, Austin, TX, USA

ACM ISBN 978-1-4503-7116-2/20/03…$15.00
https://doi.org/10.1145/3376897.3377862

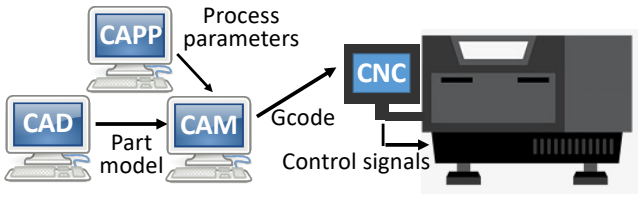


Figure 1: Traditional manufacturing automation model.

Strict timeliness and reliability requirements. Manufacturing machines expect to receive input at a constant rate and must immediately react to any local input, e.g., an operator may decide to halt the ongoing job, but the execution must terminate gracefully when doing so. For applications that run on our laptops and smartphones and rely on input from the cloud, delayed input or lack of input can lead to annoyances such as rebuffering and stalled videos. In the context of manufacturing machines, lack of input when expected is intolerable. For instance, abruptly halting a machine's execution can affect the device's mechanical parts and result in vibration marks on the manufactured parts, thus ruining them and leading to significant losses. For these reasons, even if a machine's controller runs at a edge server nearby in order to ensure timely input [19], disconnections from that controller are challenging to tolerate.

Inherent non-determinism. On electronic devices, running the same program on multiple devices will consistently yield the same results when given the same inputs, barring sources of non-determinism which vary across runs on the same device (e.g., thread scheduling and time of day). In contrast, even when the same part is manufactured on multiple machines of the same type, the output can vary significantly due to differences in their mechanical parts, e.g., one might be more worn out than the other. Consequently, when multiple parts of a product are manufactured on different machines, the end results might not compose well with each other.

Due to these unique characteristics, enabling remote control of manufacturing devices requires a fundamental rethink of how clients can leverage compute resources in the cloud and at the edge. In this paper, we first describe how support from the cloud and from the edge will need to be combined in order to enable more efficient, yet reliable, manufacturing. We then describe the research problems that need to be tackled in order to enable these benefits. We draw a contrast with several threads of prior work which have focused on harnessing remote computation in the more traditional context of supporting applications running on electronic devices.

2 BACKGROUND AND MOTIVATION

We begin with a high-level description of how manufacturing machines are used today, and a case study highlighting limitations of the current workflow.

2.1 Traditional Manufacturing Workflow

In the traditional computer-aided workflow for making a part (Figure 1), the part is first modeled in computer-aided design (CAD) software. Following this, computer-aided manufacturing (CAM) software is used to generate toolpaths, based on the CAD model and manufacturing process parameters. Between CAD and CAM, there is a process planning step that helps determine key parameters used in CAM; it is sometimes achieved with the help of computer-aided process planning (CAPP) software. The toolpath outputted by CAM is post processed to G-code, which contains *high-level* commands for manufacturing the part. An example high-level command is: travel along a circular path at a specified maximum speed.

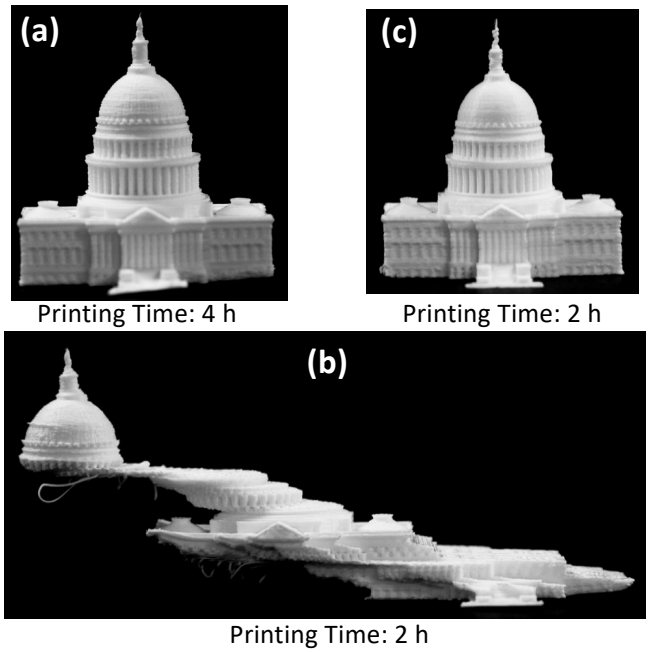


Figure 2: Prints of scale model of US Capitol (52.5 mm height) using: (a) the default CNC controller at standard printing speed; (b) the default CNC controller at higher printing speed leading to loss in quality due to excessive vibration; and (c) a more advanced controller which runs the LPFBS (vibration compensation) algorithm to drive the printer at a faster rate without loss in quality.

The component of interest in this paper is the last one involved in this workflow: computer numerical control (CNC) of a manufacturing machine. Unlike the other components, which are used offline (i.e., prior to when production of the part begins), CNC drives the execution of the machine as it produces the part. CNC turns high-level G-code commands into *low-level* commands that control the motors of the manufacturing machine – be it a 3D printer, laser cutter, milling machine, or other — to produce the desired part. For instance, the CNC breaks down the high-level command to travel along a circular path into specific details of the movement of the machine at each millisecond time interval.

2.2 Case Study: 3D Printing

The primary constraint of the status quo is that CNC runs locally on the manufacturing machine. To appreciate the poor tradeoff between quality and manufacturing introduced by this constraint, let us consider the CNC of a 3D printer.

First, *manufacturing machines like 3D printers are fairly slow*, requiring on the order of hours to complete production of any part. Therefore, when one needs to manufacture a product composed of several parts or several copies of the same product, it is desirable

that production be distributed across many machines. Buying many manufacturing machines may be expensive for a single entity, but one can leverage manufacturing-as-a-service providers (e.g., Xometry [1] and makexyx [2]), which connect those who have idle devices with others who are looking to cost-effectively manufacture parts.

However, a key challenge in distributing the production of a product's parts across multiple machines is that there can be **significant variance across machines**, yet each machine is controlled by its own locally running CNC algorithm. Even if all the machines used were of the same "type" (e.g., same make and model, and devices from the same year), prior work has shown that there can be significant differences across machines [18], e.g., due to different amounts of wear and tear. So, when production of a product's parts is spread across multiple machines, these parts might not compose well with each other. In other words, the quality of the end-product will be significantly worse than what one would obtain by manufacturing all parts on the same machine.

Second, even if the product being manufactured comprises a single part, **the hardware resources available on a machine intrinsically limit the capabilities of the CNC algorithm**. For example, in prior work, one of the co-authors has developed an advanced control algorithm – limited-preview filtered B-spline (LPFBS) – to drive 3D printers at faster speeds while minimizing the impact on product quality [8]. The high-level intuition underlying the LPFBS approach is that, rather than driving the 3D printer at a fixed speed, the controller uses a dynamic model of the device to predict the device's impending vibrations and adaptively varies the speed of execution to avoid the vibrations. Figure 2 shows an example where use of the advanced LPFBS controller halved production time compared to the default controller; running the default CNC controller at twice the standard printing speed results in severely degraded quality.

The resource requirements of the LPFBS algorithm, however, exceed those typically available on many desktop 3D printers. For example, the ATMega2560 chip, which is common on desktop 3D printers, has a 16 MHz processor with 8 KB SRAM. These capabilities are insufficient to support the execution of the LPFBS algorithm, which needs at least a 1.2 GHz CPU and 9 MB of memory. Even as device capabilities improve over time, historical trends indicate that the resource requirements of the most advanced controllers available will also continue to increase, thus rendering many devices incapable of benefiting from them.

3 REMOTE CONTROL OF MANUFACTURING

Given the limitations of running CNC locally on each manufacturing machine, a natural solution is to move this controller off the machine into a separate server. A straightforward way to do so is as depicted in Figure 3. The remote controller translates high-level commands (e.g., a G-code file) into low-level commands (e.g., stepper commands for the motors on the device), which are streamed to the device. The manufacturing machine is reduced to a "dumb client", simply buffering the received inputs and applying them at a fixed rate; this requires minimal compute/memory resources on the device.

[1] https://www.xometry.com
[2] https://www.makexyz.com

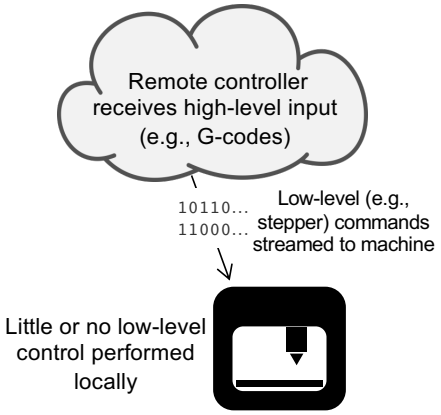


Figure 3: High-level illustration of offloading control of a manufacturing machine to a server.

3.1 Edge versus Cloud

The most natural way of hosting a machine's remote controller would be on a server within the same local area network (LAN) as the machine. Using our case study from the previous section, we argue why such an approach is neither strictly necessary nor completely sufficient.

No (or slow) feedback. In order to improve the quality versus manufacturing time tradeoff, advanced control algorithms can largely operate in feedforward mode, i.e., generate low-level commands without any feedback from the machine, as is the case with the results shown in Figure 2 using the LPFBS algorithm. While such algorithms could benefit from periodic recalibration of their model, it would suffice to obtain feedback from the machine once every few seconds. Therefore, to use advanced control algorithms, it is not vital to have sub-millisecond latencies between a machine and its remote controller; latency in the order of tens to hundreds of milliseconds, which would be commonplace if the controller were hosted in the cloud, would be tolerable in the common case.

Cost. Furthermore, not everyone who would want to benefit from off-device control of their manufacturing may be able to afford the costs of setting up an edge server to host the controller or have the expertise to do so. For example, in the case of 3D printing, a common beneficiary of remotely controlled execution will be individual users who have a single 3D printer at home. For such users, we envision advanced controllers being hosted in the cloud by service providers and a user selecting from a marketplace of such cloud controllers. In such cases, users will need to be aware of the implications with regard to both privacy (what products are being manufactured will be revealed to the service being used) and security (reliability of the machine, and perhaps even the human operator, will be at risk), as is the case when we use cloud-based services on our computers.

Collective learning. Even when one is capable of hosting a controller at the edge, sharing of information with edge controllers at other sites can benefit everyone. One edge controller's operation when driving a particular machine can help other edge controllers tailor their operation of other machines of the same type. This would require asynchronous communication among edge controllers, likely via a central service in the cloud.

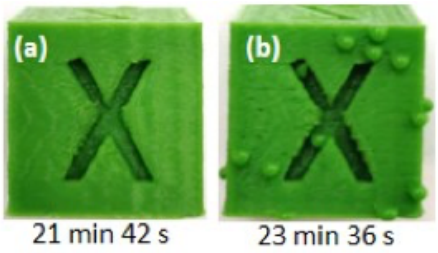


Figure 4: 3D prints of a part from a cloud controller: (a) without Internet latency-induced pauses (benchmark), and (b) with Internet latency-induced pauses. Shown below each picture is the printing time. It can be seen that the latency affects both productivity and part quality, by introducing undesirable blobs on the printed surface [17].

Independent failures. Compared to a cloud controller, an edge controller offers lower latency, lower latency variance, and higher availability. Yet, it does not solve all problems associated with remote control of manufacturing machines. Moving the controller off the device fundamentally implies that the machine and its controller are now in different failure domains and can fail independently. So, handling the unavailability of the controller becomes a must.

3.2 Research Challenges

Next, we discuss the common challenges that arise irrespective of whether the controller for a machine is run in the cloud or on an edge server.

3.2.1 Stringent Timing Requirements. **Challenge: Spikes in latency and drops in connectivity.** In settings where hosting the CNC controller at an edge server is not an option, relying on a controller in the cloud is challenging due to the lack of delay and availability guarantees over the Internet. While cloud providers today have data centers in many locations across the globe, thereby enabling customers to rent servers from a location close to their client devices, prior work has observed significant spikes in Internet latency intermittently even from clients to nearby cloud data centers [23]. Similarly, others have observed that many paths on the Internet have downtimes of over 2% [9]. After all, the Internet offers only best-effort service.

Variability in latency from the controller to the device and unpredictable availability of this communication is particularly damning for manufacturing machines. When a machine runs out of buffered low-level commands received from the controller, the machine cannot simply pause its execution, e.g., a 3D printer will continue to spew out material and inertia of the motors cannot be halted instantaneously. Figure 4 depicts an example of the adverse effects that result due to these factors.

Problems with strawman solutions. One could consider several approaches for avoiding such ill-effects, none of which are ideal.

- First, if one wishes to pause a machine's execution when it runs out of buffered commands, bringing the machine to a graceful halt must be preemptively initiated when buffer occupancy is low, in contrast to electronic devices where execution is halted once the buffer runs out. Not only might this lead to unnecessary slowdowns, but simply halting production when the remote controller is unavailable hurts productivity.
- Alternatively, one could increase the number of commands buffered on the device – in the limit, precomputing all low-level commands – so as to increase the machine's ability to cope with latency spikes and temporary losses of connectivity. Compute low-level commands far in advance is, however, not desirable; as discussed earlier, advanced control algorithms like LPFBS can benefit from feedback from the device.
- Even if one had the budget and knowledge to sidestep the Internet's vagaries by hosting the controller on an edge server, disconnections from the remote controller will still be feasible and must be tolerated.

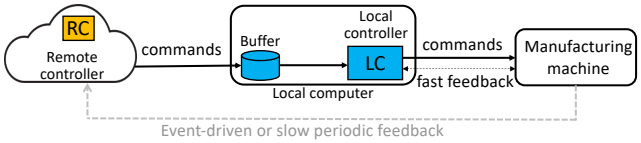


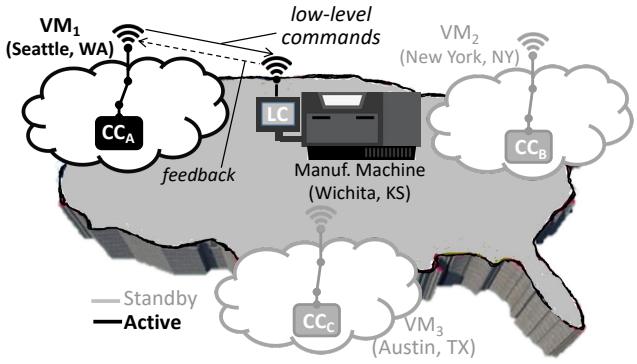
Figure 5: Combining use of an off-device controller with a local controller which assumes control when either connectivity to the remote controller is lost or feedback must be immediately handled (e.g., power down the machine).

Solution: Local *and* remote control. To leverage a off-device controller while being tolerant to losses of connectivity and latency spikes, we envision the approach depicted in Figure 5. Instead of the machine acting as a dumb client which simply executes low-level commands, we foresee the need for a local controller which can take over from the remote controller when buffered commands run out. The local controller would clearly be incapable of running the same algorithm as the remote controller; this is the reason why we seek to move the machine's control off the device in the first place. However, falling back to the local controller and switching to a lower speed of operation is better than degrading the quality of the part being manufactured or simply halting production. Control of the device can be handed back to the remote controller once connectivity to it is restored.

Why is handoff of control hard? There are two primary challenges in realizing this synergy between the local and remote controllers.

- On first glance, this handoff of execution back and forth between the two controllers may appear similar to prior solutions for dynamic partitioning of applications between mobile devices and the cloud [3–5, 12]. However, the setting here differs in one key property: all of these prior efforts assume that both the local and remote instances of the application are running the same algorithm, and handing off from one to the other is guided by the need to speed up computation or reduce energy consumption. In contrast, many manufacturing machines are fundamentally incapable of running advanced control algorithms, and therefore, the algorithms underlying the local and remote controllers will differ.

  This crucial difference calls for a new approach that one could use to orchestrate execution between the local and remote



VM = virtual machine; CC = cloud controller; LC = local computer

Figure 6: Run redundant cloud controllers in multiple regions to cope with losses in connectivity. The challenge is in migrating the state built up based on machine feedback.

controllers of a manufacturing machine. A solution that is agnostic to the control algorithm is desirable, but it must account for differences in the algorithms used by the two controllers.

- Handing off control of a machine from the remote controller to the local controller and back must be done with care because the switch between the two must be performed gracefully. For instance, if the machine were to abruptly transition from the high acceleration that the remote controller was running it at to a lower speed of execution that the local controller is capable of driving, this will put the mechanical parts at risk of failure.

  Therefore, the control algorithm that runs off-device must allow for potential rendezvous points to the lower fidelity [16] algorithm (i.e., one which runs the device at a lower speed in order to maintain quality) that the machine can run locally. Abstractly, the remote controller must ensure at any point in time that, after the machine has executed the last N low-level commands received from it, there exists a potential path of execution for the local controller to seamlessly transition to a sequence of commands that it is capable of generating. While this property is trivially satisfiable if both controllers run the same algorithm, the open question here is whether it is feasible to do so without curtailing the benefits of the advanced control algorithms that we seek to run in a cloud/edge server.

3.2.2 Stateful Controller. As discussed above, we envision switching a machine's controller from one running remotely to one running locally when the machine's connectivity to the remote controller is lost. While having the local controller drive the machine ensures safe operation of the device, running in a degraded mode for the remainder of the machine's ongoing execution is undesirable. This will result in elongated production times.

Leveraging redundancy. In cases where the remote controller runs in the cloud and the machine's connectivity to that controller is lost, we envision that the local controller can hand off to a controller in a different cloud region. As shown in Figure 6, this will require redundant controllers to be running in standby mode in several locations. Public cloud providers make it easy to do so as they have data centers in many regions spread across the globe. Even when a machine loses connectivity to one data center, it might still be able to communicate with controllers in other data centers. Switching to one of them will help ensure that the loss of productivity when the local controller is in-charge will be temporary.

Realizing this vision is, however, far from straightforward if we wish to support advanced control algorithms which could benefit from slow feedback from the device. For example, it is desirable for the LPFBS algorithm [8] described previously to collect measurements of on-device vibrations, so as to help improve its predictions of impending vibrations. At any point in time, the cloud controller would have accumulated some state based on such feedback. Moving control of a device from one cloud region to another requires transfer of this state.

From cloud to device and back. Again, at first glance, handoff between controllers in multiple cloud regions appears similar to the state handoff performed in mobile edge computing. There too, when a mobile user moves from one location to another, the application running in the cloud has to transfer state across edge servers (e.g., cloudlets, micro-clouds, base stations, etc.) [14, 21, 22].

However, a key difference in the two settings renders prior solutions unusable in the manufacturing context. In the mobile edge computing scenario, the service always runs remotely, and as the user moves, the service migrates state from one edge server to another. In contrast, control of a manufacturing machine cannot directly be passed from one cloud region to another. Such a handoff is necessary primarily when the machine loses connectivity to the cloud region in which the controller currently driving it is hosted. Unlike how predictions of the user's next location can be used in the mobile computing setting to determine when to migrate service state, connectivity disruptions are unpredictable. If the handoff between cloud regions is initiated once loss of connectivity to the device is observed, the local controller must take over control of the machine while this handoff is in progress.

Therefore, the open question here is: how to bootstrap a backup cloud controller when it takes over? To seamlessly resume execution, the new controller must be made aware of the state that the previous cloud controller had accumulated based on the feedback it had received from the machine. But, this state must be updated to account for the progress made by the local controller before it handed off control to the backup cloud controller. The initial state on the backup cloud controller must, therefore, be initialized by carefully combining inputs received from both the original cloud controller and the local controller.

3.2.3 In-Process Profiling. We have thus far discussed challenges in controlling a single manufacturing device from a cloud or edge server. Coordinating the execution of multiple machines (e.g., to print multiple parts of a product) is challenging too.

Like distributed data processing, but not quite. In this case, the analogy we draw is to the large body of prior work on distributed data processing frameworks such as MapReduce [6], Spark [24], etc. In these frameworks, typically a central coordinator orchestrates the execution of many workers, assigning a portion of the overall work to each worker. The compilation of outputs from all workers represents the overall output.

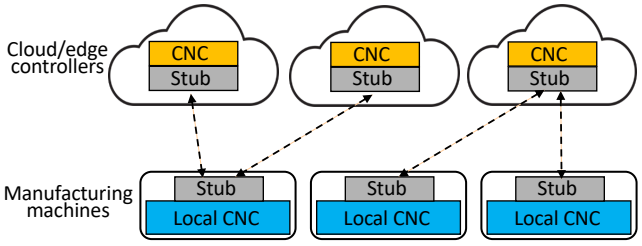


Figure 7: Envisioned system architecture.

Similarly, to manufacture a product composed of multiple parts, an off-device controller overseeing many machines can assign the production of a subset of parts to each machine. The final product can be constructed by composing the individual parts.

The key difference is as follows: while processing the same data will yield the same results irrespective of the computer on which this is executed, the production of a part can vary significantly across machines. So, the remote controller cannot simply drive the execution of each machine independently. Instead, to ensure that the individual parts do compose well with each other, the remote controller must take into account the feedback it receives from each machine to not simply fine-tune the execution of *that* machine, but also inform the execution of *other* machines used to manufacture other parts of the same product. In other words, unlike in distributed data processing, where processing of each data partition can be performed independently in each stage of a job, the unpredictability of manufacturing machines makes cross-device information sharing mandatory.

4 CONCLUSION

In summary, our goal in this paper was to make the systems research community aware of the challenges that exist in the new emerging domain of smart manufacturing. While the motivating factors for utilizing support from the cloud or the edge are similar in the manufacturing context to those in traditional software services and applications, we argued that several existing techniques – such as offloading computation, migrating service state across compute units, and coordinating execution across devices – need to be rethought. This presents an exciting new opportunity for systems researchers.

As shown in Figure 7, we foresee the ideal end-goal here is to develop a middleware that spans manufacturing machines and the cloud/edge, handling handoffs from servers to the machine and back, state migration, coordination based on in-process profiling, etc. This would entail offering client-side and server-side stubs that abstract away all of these details from the CNC modules that run on- and off-device. Advanced control algorithms for driving the operation of manufacturing machines can then evolve independently, freeing researchers who develop such algorithms from having to worry about any of the aforementioned complexities.

5 ACKNOWLEDGEMENTS

We thank the anonymous HotMobile reviewers and our shepherd, Mahadev Satyanarayanan, for their feedback on earlier drafts of this paper. This work was supported in part by the NSF under grant CNS-1563849.

REFERENCES

[1] Yusuf Altintas, Alexander Verl, Christian Brecher, Luis Uriarte, and Günther Pritschow. 2011. Machine tool feed drives. *CIRP annals* 60, 2 (2011).
[2] Marco Annunziata. 2019. Manufacturing-As-A-Service Platforms: The New Efficiency Revolution. https://www.forbes.com/sites/marcoannunziata/2019/05/13/manufacturing-as-a-service-platforms-the-new-efficiency-revolution/.
[3] Rajesh Krishna Balan, Mahadev Satyanarayanan, So Young Park, and Tadashi Okoshi. 2003. Tactics-based remote execution for mobile computing. In *MobiSys*.
[4] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. 2011. Clonecloud: Elastic execution between mobile device and cloud. In *EuroSys*.
[5] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. 2010. MAUI: Making smartphones last longer with code offload. In *MobiSys*.
[6] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified Data Processing on Large Clusters. *CACM* 51, 1 (2008).
[7] Nancy Diaz, Moneer Helu, Stephen Jayanathan, Yifen Chen, Arpad Horvath, and David Dornfeld. 2010. Environmental analysis of milling machine tool use in various manufacturing environments. In *IEEE International Symposium on Sustainable Systems and Technology*.
[8] Molong Duan, Deokkyun Yoon, and Chinedum E Okwudire. 2018. A limited-preview filtered B-spline approach to tracking control–With application to vibration-induced error compensation of a 3D printer. *Mechatronics* 56 (2018).
[9] Osama Haq, Mamoon Raja, and Fahad R Dogar. 2017. Measuring and improving the reliability of wide-area cloud paths. In *WWW*.
[10] Will Jacobsen. 2017. How to reduce the risk of component obsolescence. https://www.controldesign.com/vendornews/2017/how-to-reduce-the-risk-of-component-obsolescence/.
[11] Siddharth Kale, Nattasit Dancholvichit, and Chinedum Okwudire. 2014. Comparative LCA of a linear motor and hybrid feed drive under high cutting loads. *Procedia CIRP* 14 (2014).
[12] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. 2012. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *INFOCOM*.
[13] Bo-Hu Li, Lin Zhang, Shi-Long Wang, Fei Tao, JW Cao, XD Jiang, Xiao Song, and XD Chai. 2010. Cloud manufacturing: a new service-oriented networked manufacturing model. *Computer integrated manufacturing systems* 16, 1 (2010).
[14] Andrew Machen, Shiqiang Wang, Kin K Leung, Bong Jun Ko, and Theodoros Salonidis. 2017. Live service migration in mobile edge clouds. *IEEE Wireless Communications* 25, 1 (2017).
[15] Mostafa G Mehrabi, A Galip Ulsoy, and Yoram Koren. 2000. Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing* 11, 4 (2000).
[16] Brian Noble, Mahadev Satyanarayanan, Dushyanth Narayanan, Eric Tilton, Jason Flinn, and Kevin Walker. 1997. Agile application-aware adaptation for mobility. (1997).
[17] Chinedum Okwudire, Sharankumar Huggi, Sagar Supe, Chengyang Huang, and Bowen Zeng. 2018. Low-Level Control of 3D Printers from the Cloud: A Step toward 3D Printer Control as a Service. *Inventions* 3, 3 (2018).
[18] S Sartori and GX Zhang. 1995. Geometric error measurement and compensation of machines. *CIRP annals* 44, 2 (1995).
[19] Mahadev Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017).
[20] K Schischke, E Hohwieler, R Feitscher, J König, S Kreuschner, P Wilpert, and NF Nissen. 2012. Energy-Using Product Group Analysis–Lot 5 Machine tools and related machinery. *Fraunhofer Institute, Berlin* (2012).
[21] Shiqiang Wang, Rahul Urgaonkar, Ting He, Murtaza Zafer, Kevin Chan, and Kin K Leung. 2014. Mobility-induced service migration in mobile micro-clouds. In *MILCOM*.
[22] Shiqiang Wang, Rahul Urgaonkar, Murtaza Zafer, Ting He, Kevin Chan, and Kin K Leung. 2015. Dynamic service migration in mobile edge-clouds. In *IFIP Networking*.
[23] Zhe Wu, Curtis Yu, and Harsha V Madhyastha. 2015. CosTLO: Cost-effective redundancy for lower latency variance on cloud storage services. In *NSDI*.
[24] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*.
[25] Peter Zelinski. 1998. Choosing a Retrofit CNC. *Modern Machine Shop* 70, 12 (1998).