A One Year Retrospective on a MOOC in Parallel, Concurrent, and Distributed Programming in Java

Vivek Sarkar
Georgia Institute of Technology
vsarkar@gatech.edu

Max Grossman
Rice University
max.grossman@rice.edu

Zoran Budimlic
Rice University
zoran@rice.edu

Shams Imam
Two Sigma
shams.imam@twosigma.com

***Abstract*—Much progress has been made on integrating parallel programming into the core Computer Science curriculum of top-tier universities in the United States. For example, "COMP 322: Introduction to Parallel Programming" at Rice University is a required course for all undergraduate students pursuing a bachelors degree. It teaches a wide range of parallel programming paradigms, from task-parallel to SPMD to actor-based programming.**

However, courses like COMP 322 do little to support members of the Computer Science community that need to develop these skills but who are not currently enrolled in a four-year program with parallel programming in the curriculum. This group includes (1) working professionals, (2) students at USA universities without parallel programming courses, or (3) students in countries other than the USA without access to a parallel programming course.

To serve these groups, Rice University launched the "Parallel, Concurrent, and Distributed Programming in Java" Coursera specialization on July 31, 2017. In 2017, the authors of that specialization also wrote an experiences paper about launching the specialization.

In this paper, the sequel to our previous publication, we look back at the first year of the Coursera specialization. In particular, we ask the following questions: (1) how did our assumptions about the student body for this course hold up?, (2) how has the course changed since launch?, and (3) what can we learn about how students are progressing through the specialization from Coursera's built-in analytics?

***Index Terms*—parallel programming, pedagogy, concurrent, distributed, online, MOOC, Coursera**

I. Introduction

Massive Open Online Courses, or MOOCs for short, have proven to be an effective strategy for scaling advanced and technical education beyond the walls of four-year universities. MOOCs have seen success in everything from data science [6] to finance [3] to social psychology [15].

Arguably the field that has seen the most MOOCs generated is Computer Science. Today, online students can learn specific languages (e.g. R, Python, Java), algorithms, cryptography, machine learning, and even topics in quantum computing from courses offered by top professors from well-known institutions. One would be hard-pressed to find an area of Computer Science with no representation in the set of available MOOCs.

However, up until 2017, there were relatively few offerings of online parallel programming courses [11, 12] and most of them were narrow in scope. They included:

1) "Heterogeneous Parallel Programming" [18]: Offered by Professor Wen-Mei Hwu of the University of Illinois Urbana-Champaign on the Coursera platform, this course focuses on teaching the essential parallel programming concepts for programming multi-core CPUs and GPUs using OpenCL or CUDA.
2) Udacity CUDA course [9]: With a similar focus to Professor Hwu's course, this offering on the Udacity platform focuses specifically on teaching how to program with CUDA without much discussion of the fundamentals of parallelism.
3) "Clouds, Distributed Systems, and Networking" specialization [2]: A suite of courses offered by the University of Illinois Urbana-Champaign on the Coursera platform, that focuses on topics in cloud computing. Unlike previous courses, this course teaches both abstract concepts and practical aspects of cloud computing.
4) "Parallel programming" [8]: This course teaches shared-memory parallelism in the Scala programming language. Students are asked to write correct parallel programs, but are not graded on the performance of their implementations.

While all of the above courses are of the highest quality, their scope is generally limited: CUDA programming, OpenCL programming, cloud computing, and parallel programming in Scala. The MOOC community lacked a general-purpose introduction to parallel computing.

However, on July 31, 2017, Rice University launched the "Parallel, Concurrent, and Distributed Programming in Java" specialization on the Coursera platform [1] (PCDP for short). This specialization included three courses: "Parallel Programming in Java", "Concurrent Programming in Java", and "Distributed Programming in Java".

"Parallel Programming in Java" focuses on how to create and coordinate parallelism, with topics in both task and dataflow parallelism. "Concurrent Programming in Java" builds on the foundation provided in the Parallel course, and focuses on managing concurrent accesses to data structures shared by multiple threads. Finally, "Distributed Programming in Java" emphasizes topics in multi-process parallelism, such as big data frameworks like Apache Spark or high-performance computing frameworks like MPI.

Each of these courses was split into multiple weeks, and each week included multiple lecture videos, readings, and a

hands-on programming exercise. In designing these courses, it was important to acknowledge the expected mix of students – while some may be interested in a purely academic sense, many would expect to gain hands-on experience that could then be applied in the workforce. More details about the PCDP specialization can be found in the original experiences paper published on it, "Preparing an Online Java Parallel Computing Course" [14].

It has now been over a year since the launch of this specialization, with over $16,000$ all-time active learners across all courses. To our knowledge, there has been no new launches of courses in general-purpose concurrent and parallel programming since the launch of the PCDP specialization. As a retrospective, we explore the following four questions in this paper:

1) How did our assumptions about the student body for this course hold up?
2) How has the course changed since launch?
3) What can we learn about how students are progressing through the specialization from Coursera's built-in analytics?

The insights and experiences shared in this paper can serve as useful guidance for other HPC educators looking to launch their own online courses, or looking to use material from the Coursera specialization in their in-person offerings.

II. The Learners

Coursera defines active learners as "unique enrolled learners who viewed a reading or discussion, began watching a video, or began an assessment; includes both mobile and web users". In total, we have received $8,390$ active learners in the Parallel course, $4,316$ in the Concurrent course, and $3,736$ in the Distributed course as of August 22, 2018. We see course completion rates of 42.3% in Parallel, 53.2% in Concurrent, and 49.4% in Distributed (see Section IV for more details). While we do not have the ability to uniquely identify learner enrollment across courses, we suspect that there is a strong overlap of learner populations across all three courses.

In the paper describing the PCDP Coursera specialization [14], we stated that we had two expectations about the type of learners for which this specialization would be most valuable:

1) Learners who were already in the workforce, and were looking to broaden their skill sets.
2) Learners who were full-time or part-time students, but lacked access to parallel programming courses.

Fortunately, the Coursera platform includes built-in analytics that can help instructors learn about the make up of learners in their courses. Using these analytics, we can demonstrate that our expectations aligned well with the actual demographics of the learner population.

A. Employment and Part/Full-Time Students

Figure 1 reports the percentage of learners who are (1) employed full-time, or (2) a part/full-time student, in each course. Clearly, both groups are well-represented. At least two-thirds of each course is composed of learners who are also working full-time, indicating that our courses are serving the stated purpose of educating professional software engineers who no longer have access to the educational resources of a four-year university. Note that we have no way to identify the particular industry a learner is employed in – a learner may be employed full-time in a role other than as a software engineer.

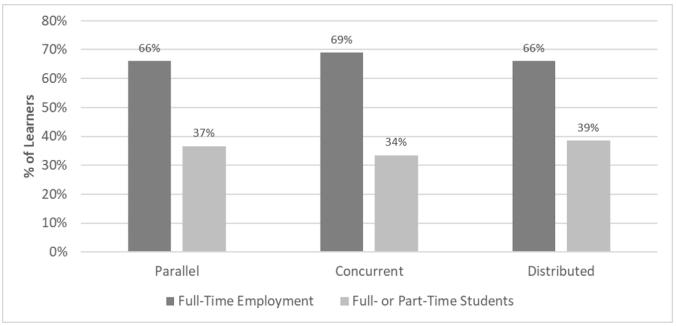

Fig. 1. Percentage of learners in each course that self-reported as being either employed full-time or a part/full-time student. Note that there may be overlap in these groups. For example, a single respondent could report themselves as both employed full time and a part-time student.

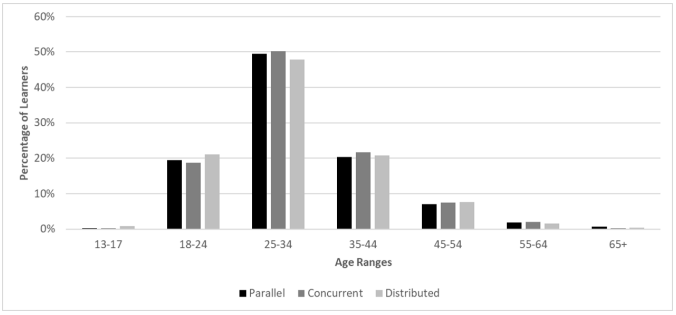

Fig. 2. Percentage of learners in each course within each reporting age range.

Additionally, greater than one-third of each course's learners also identify as part-time or full-time students, suggesting that even students currently enrolled at educational institutions lack access to parallel computing curriculum. Note that these responses do not identify the type of institution a student is enrolled at (e.g. high school, two-year university, four-year university, etc).

B. Learner Age Distribution

It is also interesting to consider the age distribution of learners in each course. Figure 2 shows this distribution across all three courses. At a glance, we can see that (1) the distribution of learner ages is approximately the same across all courses, and (2) the bulk of learners fall into the age buckets $18-24$, $35-44$, and $45-54$. By far the largest group is $25-34$ years old, which mirrors the two-thirds of the student body of each course who self-report as being employed full-time. Hence, we can postulate that a common profile of a learner in these courses is someone working full-time who has either completed university or never attended, but who is

trying to leverage online materials to improve understanding of the subject matter to either (1) advance in their current industry, or (2) move into the software industry.

C. Learner Geographic Distribution

MOOCs have the potential to attract students from different countries and cultural backgrounds. In our previous paper, we explicitly called out the desire to reach an international audience to make these material more accessible to learners in developing and emerging nations. While the material is only available in English, Table I shows the extent to which learners from a variety of different countries have engaged with the course, with up to 135 countries represented for a given course.

TABLE I
TOTAL NUMBER OF COUNTRIES WITH ACTIVE LEARNERS, PER COURSE.

Course	# of Countries
Parallel	135
Concurrent	123
Distributed	129

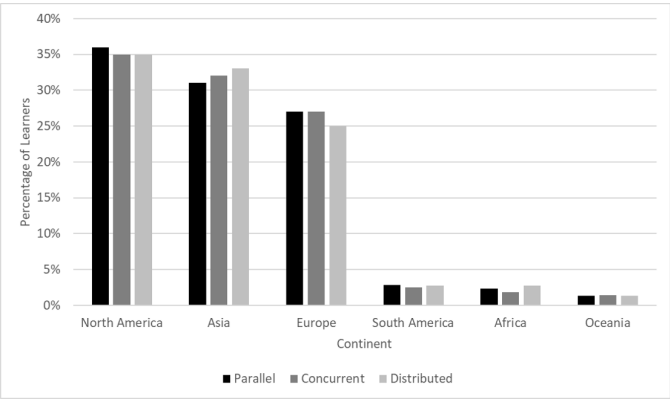

Fig. 3. Percentage of learners from each continent.

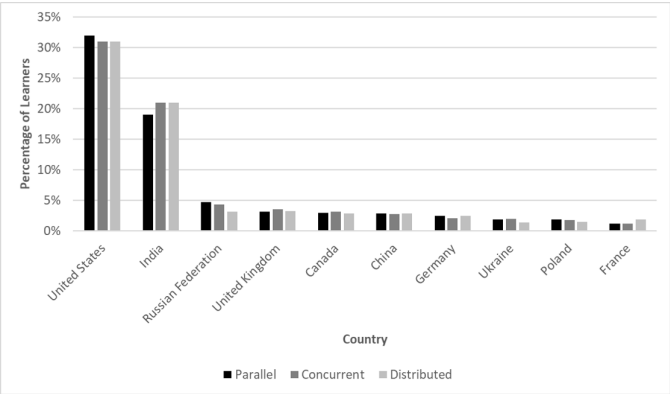

Fig. 4. Percentage of learners from the top ten represented countries.

Figure 3 shows the percentage of active learners from each continent, while Figure 4 shows the percentage of active learners from the ten most common countries. Again, distributions across the three courses are fairly similar, with learners spread throughout the world, most commonly from North America, Asia, or Europe.

D. Learner Gender Distributions

Table II shows the percentage of men and women learners in each course. On average, Coursera courses are 62% male and 38% female, illustrating that our learners are more predominantly male than the average. Still, a survey of HarvardX and MITx courses reported some courses with similar ratios (e.g. 13% female and 87% male for a course on circuits and electronics) [5].

TABLE II
PERCENTAGE OF MEN AND WOMEN LEARNERS.

Course	% Men	% Women
Parallel	91.0%	9.0%
Concurrent	91.3%	8.7%
Distributed	90.0%	10.0%
Coursera (all courses)	62.0%	38.0%

E. Learner Reviews

Coursera also allows learners to rate courses that they take, and provide "stories" about how they are using these courses in their careers. Table III summarizes the ratings received by each course, all of which have a mean of well above 4 out of 5 stars.

TABLE III
LEARNER RATINGS, BROKEN DOWN BY COURSE.

Course	Average Rating (out of 5)	Number of Ratings
Parallel	4.5	370
Concurrent	4.5	230
Distributed	4.4	130

Many of the learner stories shared reflect trends in the aggregate metrics from above, underscoring the diversity of the student population with an added personal touch:

1) "I'm a software engineer in Uruguay. Thanks you were very helpful and clear explaining the course. I hope I can make good use of the things a learned here at work. Greetings!"
2) "I am ... software developer with 2 years experience and I wanted to tell you that this is most useful specialization course material I have found."
3) "I am a PhD student at Nazarbayev University (Astana, Kazakhstan), department of Electrical and Computer Engineering. This course really helped me as I implemented one of my simulations in Java streams. It also directed to work more on phasers and barriers."
4) "Thank you Professor for your course. I'm 52, Italian, and I work as senior developer in a big company that offers services B2B and B2C."

F. Use of Material by On-Campus Courses

Additionally, one way the original PCDP paper foresaw the course material being used by learners was through on-campus

courses, taught by instructors that leveraged the online course as a springboard for their own curriculum.

While the open nature of the material means it is difficult to track all instances of this phenomenon (i.e., an instructor may simply re-use course materials in their own course without necessarily having contacted us), we know of at least one existing case. Professor Dennis Cosgrove of Washington University in St. Louis worked with the designers of the PCDP specialization to leverage some of the parallel computing infrastructure, readings, and lecture videos in his on-campus course "CSE 231: Intro to Parallel and Concurrent Programming".

We hope that other instructors will be able to continue to leverage the work put into the online course and supporting infrastructure in developing and expanding on-campus offerings.

III. Course Changes Since Launch

Changes to the course are important for the continuous improvement of the course. They allow the course to address needs of the learners by incorporating feedback. We break the changes to the PCDP specialization into four groups:

1) Changes to course material and curriculum.
2) Changes to course mini-projects.
3) Changes to the course autograder.
4) Changes to the supporting PCDP library.

A. Course Material Changes

While there have been small corrections to reading material and lecture videos (e.g. due to instructor misspeaks), the course material for all three courses in the specialization has remained relatively fixed since the launch. The most common feedback received from learners is generally concerning corrections to the subtitles added to lecture videos. The relative lack of changes to the course curricula, complimented with our relatively high ratings, attest to the learners agreeing with the course design and finding it useful.

B. Changes to Course Mini-Projects

Each course in the PCDP specialization is broken up into 4 weeks. Each week includes a hands-on mini-project where learners take a skeleton Java program and extend it in some way that illustrates concepts learned during the week. Apache Maven [16] was used as the build system for these projects, and JUnit tests were provided to learners to verify the correctness and performance of their code. These mini-projects were tested pre-launch by at least two Rice University undergraduates per mini-project, who had completed Rice's introduction to parallel programming course.

While updates to these mini-projects were few and far between, they were necessary after course launch as learners uncovered issues with the mini-project code:

1) **Parallel Mini-Project 1**: The data set sizes and number of test repeats were both increased in order to reduce inconsistent performance test results. Java's Fork/Join common pool was also replaced with an explicitly created and sized `ForkJoin` pool.
2) **Parallel Mini-Project 3**: The number of test repeats was increased and test results for only two cores were removed from the grading criteria to avoid inconsistent results.
3) **Parallel Mini-Project 4**: The initialization of test data structures was modified to ensure that if data races occurred in the provided solution, they would appear as incorrect results. Designing test cases that would be fragile to data races was one challenge of the overall mini-project development, so as to prevent learners passing tests by chance.
4) **Concurrent Mini-Project 4**: Removed percent symbols from the output of some tests, as the auto-grading system was failing to parse them.

Given that the course has been live for over a year now, that means we have only had to update mini-projects less than once every three months, despite thousands of learners progressing through these courses.

C. Changes to Course Autograder

Note that by far the greatest point of concern to date with the mini-projects has been inconsistent performance test results on the Coursera auto-grading infrastructure. While students are able to run mini-project tests locally on their personal laptops, grades are assigned based on passing correctness and performance tests being run in the Coursera auto-grading cloud. The Coursera auto-grading cloud is a containerized environment maintained by Coursera and offered to courses to use for grading programming assignments. During the design of the course, we developed a number of re-usable scripts for deploying assignments into the Coursera cloud and running their tests.

The only changes needed to these scripts were related to filtering or replacing outputs from learner or instructor code that would cause the parsing of the autograder output by the Coursera infrastructure to fail.

D. Changes to PCDP library

Much of the PCDP specialization is taught using standard Java constructs. However, a simple, pedagogic, task-parallel programming library was also designed in support of the PCDP specialization as a simpler way to get started in parallel programming. This library is called PCDPlib, and is available open source with an $EPL - v1.0$ at https://github.com/habanero-rice/PCDP.

While no changes have been made to PCDPlib by the instructors of the PCDP specialization, we have been happy to accept changes from learners, including both performance and semantics improvements to the code base. One of the hopes stated in the original PCDP paper was that by open sourcing this piece of the course infrastructure, more advanced learners who wanted to understand the inner workings of parallel runtimes would be able to do so by looking beneath parallel programming APIs they were already familiar with.

Hence, PCDPlib's design was kept deliberately simple and readable. These change requests from learners indicate that this strategy is already paying off, and that learners are leveraging the openness of our pedagogic software stack.

IV. Learner Progression

It is also interesting to consider how learners are progressing through each course in the PCDP specialization. While the built-in Coursera analytics do not offer information on the amount of time taken to complete modules, they do offer fine-grained information on where learners drop out of a given course based on which weeks and pages of a course they view. From this, we can learn useful things about the behavior of online learners.

A. Overall Completion Rates

The total number of learners reported by Coursera for a given course is defined as the "number of unique learners who enrolled in the course". The eligible learners for a course are a subset of total learners who "are eligible to complete the course, by being able to access the graded assessments". In the case of the PCDP specialization, "graded assessments" primarily refers to quizzes and mini-projects.

It is important to note that Coursera follows a subscription model – learners pay a fixed monthly fee and then may enroll in as many courses as possible. Additionally, each course in the PCDP specialization includes free and publicly accessible content, but to complete the majority of the graded assignments you must be on a Coursera subscription. Hence, eligible learners are those learners enrolled in the course who also had an active Coursera subscription at the time.

Table IV lists the total learners, eligible learners, and completion rate for eligible learners across all three courses. Previous studies have reported average completion rates around 15% for MOOCs [7, 13]. We note that our completion rates are significantly higher than those reported in previous studies.

For example, Harvard and MIT reported about 5% completion rates [5]. However, to some extent this is a misleading comparison – this 5% rate is calculated as the total number of certified learners out of the total number of registered learners. There is no cost (monetary or otherwise) to register in the HarvardX [4] or MITx [10] platforms, whereas eligible learners (on which our completion rates are based) must have a paid Coursera subscription. That monetary investment may act as an incentive to completion.

TABLE IV
Overall learner completion rate information, broken down by PCDP course. The reported % completed is a percentage of eligible learners.

Course	Total Learners	Eligible Learners	% Completed
Parallel	11,304	2,151	42.3%
Concurrent	6,629	1,211	53.2%
Distributed	6,379	929	49.4%

B. Module Completion Rates

Each course is split into multiple modules/weeks. Across all courses, Module 1 refers to an introductory module which includes general course information and a test mini-project that doesn't require students to write any code, but helps ensure their local development environments are correctly configured and gets them familiar with the Coursera autograder.

Modules 2, 3, 5, and 6 are then content modules which include video lectures, readings, and mini-projects. Module 4 focuses on a video interview between the instructors of the course and industry experts, presenting a discussion of the real-world usefulness of the practices and techniques being taught in the current course. This module aims to motivate students. Module 7 simply contains content pointing to the other courses in the specialization.

Figure 5 plots the percentage of eligible learners in each course that complete each module. It is interesting to see that the steepest drop occurs from enrolling in the course to completing the first module, as the first module does not actually include learning any content or completing any assessments. This suggests that many eligible learners register for the course but never return, or that many eligible learners simply read the material without investing time in completing the assessments.

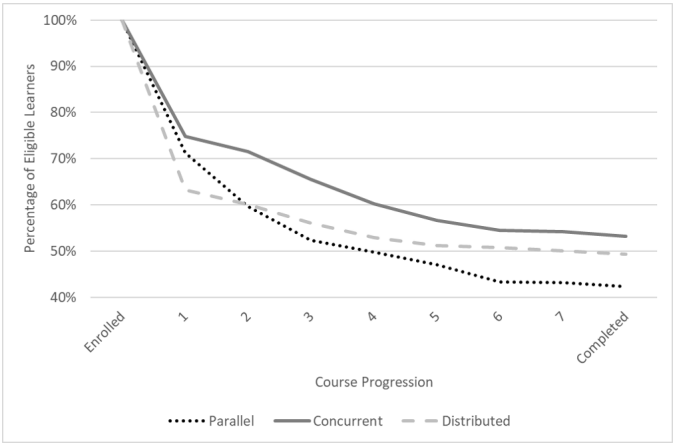

Fig. 5. Percentage of eligible learners that complete each module.

Figure 6 plots the same data as Figure 5 but instead shows the percentage of remaining learners that drop out at each module, across all courses. Visualizing the data in this way makes the significance of the drop rate before completing the first module even more apparent.

As an experiment, if we classify "engaged learners" as those learners who at least complete the introductory first module and mini-project we can compute arguably more accurate completion percentages for each course. This strategy can also partially measure learner engagement due to the course design and content. Using this metric, out of all engaged learners: 59.4% for Parallel, 71.1% for Concurrent, and 78.1% for Distributed complete the course after completing Module 1. This suggests a trend where learners become more invested

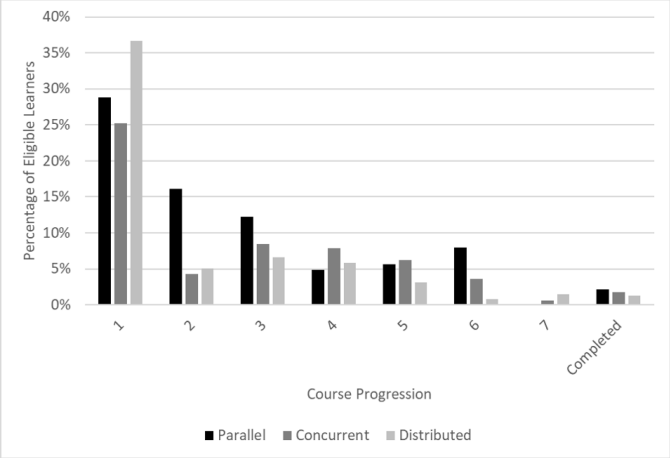

Fig. 6. Percentage of enrolled learners that dropout at each module.

and more likely to complete each course as they advance through the specialization, or as they try more advanced courses.

Coursera also exposes sub-module tracking metrics, breaking each module into five checkpoints:

1) **Started Item**: Learners starts any item in a module.
2) **Completed Item**: Learners completes any item in a module.
3) **Started Assessment**: Learner starts any assessment in a module.
4) **Completed Assessment**: Learner completes any assessment in a module.
5) **Completed Module**: Learner passes all graded assessments in a module.

Looking at these more fine grain metrics, we note that for each course the largest drop in engaged eligible learners occurs from "Completed Item" in Module 1 to "Started Assessment" in Module 1. This suggests that learners read the overview of the course but do not proceed to the test mini-project. It is unclear whether they then skip to additional readings in later modules, or exit the courses entirely and never return.

C. Module Completion Rates Breakdown by Subscription Type

One other item of interest noted during a review of the Coursera analytics was a significant difference in overall completion rates and per-module completion rates as a function of a user's subscription type. In the platform, Coursera calls out four different types of learners as subsets of all eligible learners:

1) **Paid Learners**: Personal accounts with a paid subscription.
2) **Organization Learners**: Accounts that have gained access to paid resources through membership in an organization (e.g. employees of a company with an agreement with Coursera).
3) **Finaid Learners**: Personal accounts that have applied for free access to paid resources for a specific course.

TABLE V
COURSE COMPLETION RATE AS A PERCENTAGE OF ALL ELIGIBLE LEARNERS AND SUB-GROUPS OF ELIGIBLE LEARNERS (PAID, ORGANIZATION, AND FINAID LEARNERS).

Subscription	% Completed
All Eligible Learners	42.3%
Paid Learners	46.3%
Organization Learners	37.0%
Finaid Learners	18.1%

4) **Other Access Learners**: Learners that do not fall into any of the above three buckets. It is unclear at this time what types of accounts these are, so we ignore them during this analysis.

Table V shows the final completion rates for all eligible learners and paid, organization, and finaid learners (which are each a disjoint subset of eligible learners). While paid learners see completion rates slightly higher than the average across all eligible learners, both organization and finaid accounts see lower completion rates. In particular, finaid learners see only 18% completion rates. We postulate two interpretations of this data: (1) finaid learners generally come from lower economic backgrounds and therefore may have other stressors outside of the learning platform that make it more difficult to complete the same material, and/or (2) neither organization nor finaid learners are required to invest financially in the courses and so are less motivated to complete them.

V. DISCUSSION

In previous sections of this paper, a quantitative overview of course metrics and changes was provided. In this section, we talk more anecdotally about our experiences supporting an online course for the past year.

A. Course Forums

Perhaps the most time-consuming and difficult part of maintaining a launched course is monitoring and responding to the course forums. As course enrollment grows, questions on the forums can pile up quickly. This problem is exacerbated by the fact that instructors of Coursera courses are generally kept busy by full-time jobs, and it is often difficult to reserve time daily for reading and responding to forum questions. While online platforms often promise crowd-sourced teaching assistants from students that have already completed the course, for the PCDP specialization this has not been realized to a large extent. Even when online teaching assistants are supporting the course, it is difficult to coordinate with teaching assistants who were not brought on by the instructors themselves.

In the case of the PCDP specialization, the best solution we've found to-date is keeping a part-time student employed on-campus through the digital learning center who (in addition to other duties) helps with monitoring the course forums, responding to simple requests, and highlighting more difficult requests to the course staff during weekly meetings. While it is helpful if this student has already completed the online course or similar material, in our experience it is not necessary.

While this solution works today, it naturally has long-term ramifications for the sustainability of online courses. Using a StackOverflow like model where users can vote questions and answers up or down and earn reputation points may be one way to address this issue of maintaining forums.

B. Performance Test Flakiness

Performance test flakiness is arguably the primary cause of bad learner experiences. A careful balancing act is required between setting expected speedups low enough to account for performance volatility, but high enough that students must have a reasonable solution to meet them. In our experience, a large amount of this flakiness simply derives from the lack of transparency in the Coursera auto-grading platform, the presence of virtualization, and the low levels of parallelism permitted. It is difficult to tell when volatility in test results may be due to a noisy neighbor in the Coursera platform, and it is unclear to-date how dedicated the resources assigned to a given auto-grading instance are. Additionally, auto-grading instances are limited to four CPU cores, which means that if performance scaling is expected to be sub-linear with the number of cores for certain assignments, speedups at four cores may still be marginal.

Of course, it is also unfair to blame the various course platforms for this. To some extent, running performance tests on online auto-grading platforms is an appropriation of a platform whose primary purpose and use case is for programmatic correctness verification (where performance matters to a much lesser extent).

C. Honor Code

One issue that comes up infrequently but is still worth noting is accidental honor code violations. Often, when students are requesting advice or support on the course forums they will invariably post their current solution or a code snippet. Other learners may then be able to come along and use this code snippet as a head start on the assignment. Accidental violations like these are one example of the importance of close forum monitoring – without regular monitoring of the forums, they would not be caught quickly.

In response to repeated issues like the one above, the instructors of the PCDP specialization started a dedicated e-mail account that students can use to directly and privately communicate with the course staff. While use of this account is generally discouraged (as the conversation is not searchable by the rest of the course learners), it can be useful in certain cases. In addition, it is possible in the Coursera platform for course staff to pull up the source code that learners submit to the autograder. This can be another useful avenue of communication, ensuring that both the staff and learner are looking at the same code base.

D. Engaging Learners Quickly

Section IV pointed out that learners frequently drop out of the PCDP courses after the first module, despite the fact that the first module has little content. Therefore, one important future improvement to the PCDP courses (and any other online parallel computing courses) would be to explore ways of engaging learners quickly – getting them excited about the content while keeping the barrier to entry low. This problem can be approached from a number of angles, for example: (1) by offering concrete examples of impactful applications that are only solvable using parallel computing, or (2) by pointing out the value of skills in parallel computing on the job market. However, today this is not something being accomplished well by the course introduction.

In our experience with on-campus classes, one way to engage learners is through visual depictions of what parallel and high-performance computing can do. For example, in lectures a particle simulation will be shown running sequentially and in parallel. The combination of an eye-grabbing simulation and a visual representation of how parallel computing helps can be compelling for learners.

E. Maintaining Material Long-Term

Probably the largest unanswered question for the PCDP specialization is its longevity – how long can an online course be maintained at a steady state given only part-time staff? How frequently are content updates needed to ensure the curriculum remains relevant and useful for students?

For example, in one assignment students use Apache Spark [17] to implement the PageRank algorithm. While Apache Spark is a popular and useful framework right now, will that be the case in 1 year? In 5 years? If not, when is the appropriate time to update that content? How will the staff decide what to update it to?

Of course, as pointed out previously, the recurring time investment needed to appropriately monitor the course forums is another challenge for maintaining an online course in the long term.

VI. Conclusions

At the start of this paper we posed three questions:

1) How did our assumptions about the student body for this course hold up?
2) How has the course changed since launch?
3) What can we learn about how students are progressing through the specialization from Coursera's built-in analytics?

Throughout this paper we have offered quantitative and qualitative perspectives on all of these questions, generally along several axes.

In general, our expectations for the composition of the student body have held up. The vast majority of current students are already working full-time and using the PCDP specialization to supplement what existing education they have. At the same time, approximately a third of current students are part- or full-time students that are using the specialization to supplement their ongoing education, possibly because their institutions lack the resources to offer courses in parallel or high-performance computing or they lack the time to invest in a full course on the subject. We have also seen the

adoption of this material by on-campus classes, though to our knowledge there are no on-campus classes built wholesale on top of the Coursera specialization.

For the most part, there have been few changes to the course material, mini-projects, or infrastructure since the course launch over a year ago. Most changes have been small grammatical or subtitle fixes that do not significantly impact the majority of learners' experiences.

Studying Coursera's built-in metrics yielded a few interesting (but perhaps unsurprising) metrics about learners' progression through the courses:

1) Learners on Coursera subscriptions are eager to enroll in the courses, but often disappear after reading the course overview. Perhaps these enrollments are serving as reminders and these learners will return later, but today we see a drastic drop in participation even before reaching the beginning of the real content in the course.
2) Learners that are financially invested in the courses see much higher completion rates (46.3%) than learners that gain free access to courses either through membership in an organizational account (completion rate of 37.0%) or through Coursera's financial aid system (completion rate of 18.1%).
3) The vast majority of learners that enroll in one of the PCDP's specialization's courses are not subscribed to Coursera and therefore are not able to access most of the graded, hands-on assessments. For example, out of the 11,304 total learners for the Parallel course only 2,151 are eligible for receiving a certificate at completion of the course.

When comparing to past publications on online courses [5], we generally see (1) higher completion rates, and (2) a more imbalanced learner population in terms of gender.

In general though, the PCDP specialization content is holding up to the test of time and serving the under-served groups of our broad Computer Science community that it was originally intended for.

Acknowledgment

We would like to thank Coursera, the Rice Center for Digital Learning and Scholarship, the teaching staff for COMP 322, and Rice University for their support in developing the PCDP Specialization. In particular, we would like to acknowledge the significant support and advice provided by Seth Tyger, Annette Howe, Chong Zhou, Jason Hwang, and Frank Chen.

References

[1] Coursera. Parallel, Concurrent, and Distributed Programming in Java Specialization. https://www.coursera.org/specializations/pcdp.

[2] Farivar, Reza and Singla, Ankit and Gupta, Indranil and Godfrey, P. Brighten and Campbell, Roy H. Clouds, Distributed Systems, and Networking. https://www.coursera.org/specializations/cloud-computing.

[3] Gautam Kaul. Introduction to Finance: Valuation and Investing Specialization. https://www.coursera.org/specializations/valuation-investment.

[4] Harvard University, edX. HarvardX: Free online courses from Harvard University. https://www.edx.org/school/harvardx.

[5] A. Ho, J. Reich, S. Nesterko, D. Seaton, T. Mullaney, J. Waldo, and I. Chuang. HarvardX and MITx: The first year of open online courses, fall 2012-summer 2013. 2014.

[6] Jeff Leek, Roger D. Peng, Brian Caffo. The Data Scientist's Toolbox. https://www.coursera.org/learn/data-scientists-tools.

[7] Katy Jordan. MOOC Completion Rates: The Data. http://www.katyjordan.com/MOOCproject.html.

[8] Kuncak, Viktor and Prokopec, Aleksandar. Parallel Programming. https://www.coursera.org/learn/parprog1.

[9] Luebke, David and Owens, John and Roberts, Mike and Lee, Cheng-Han. Intro to Parallel Programming. https://www.udacity.com/course/intro-to-parallel-programming--cs344.

[10] MIT, edX. MITx: Free online courses from Massachusetts Institute of Technology. https://www.edx.org/school/mitx.

[11] MOOC List. Concurrent Programming MOOCs and Free Online Courses. https://www.mooc-list.com/tags/concurrent-programming.

[12] MOOC List. Parallel Programming MOOCs and Free Online Courses. https://www.mooc-list.com/tags/parallel-programming.

[13] F. O. Onah, J. E. Sinclair, and R. Boyatt. Dropout Rates of Massive Open Online Courses: Behavioural Patterns. 2014.

[14] V. Sarkar, M. Grossman, Z. Budimlić, and S. Imam. Preparing an Online Java Parallel Computing Course. In *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*, pages 360–366. IEEE, 2017.

[15] Scott Plous. Social Psychology. https://www.coursera.org/course/socialpsychology.

[16] The Apache Software Foundation. Apache Maven. https://maven.apache.org/.

[17] The Apache Software Foundation. Apache Spark. https://spark.apache.org/.

[18] Wen-mei W. Hwu. Heterogeneous Parallel Programming. http://academictorrents.com/details/8903d0871c652b96c7b29db738cea76902d65888.