2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS) | 979-8-3503-1948-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/MASCOTS59514.2023.10387539

# License Forecasting and Scheduling for HPC

Ahmed Burak Gulhan, Gulsum Gudukbay Akbulut, Amit Amritkar, Jack Sampson, Vasant Honovar,
Adam Focht, Chuck Pavloski, Mahmut Kandemir
The Pennsylvania State University, PA, USA
{gulhan, gulsum, amit, jms1257, vuh14, abf123, cfp102, mtk2}@psu.edu

***Abstract*—This work focuses on *forecasting* future license usage for high-performance computing environments and using such predictions to improve the effectiveness of job scheduling. Specifically, we propose a model that carries out both short-term and long-term license usage forecasting and a method of using forecasts to improve job scheduling. Our long-term forecasting model achieves a Mean Absolute Percentage Error (MAPE) as low as 0.26 for a 12-month forecast of daily peak license usage. Our job scheduling experimental results also indicate that wasted work from jobs with insufficient licenses can be reduced by up to 92% *without* increasing the average license-using job completion times, during periods of high license usage, with our proposed license-aware scheduler.**

***Index Terms*—Software-as-a-Service (SaaS), High-Performance Computing (HPC), License Management, Machine Learning.**

## I. Introduction

The use of Software-as-a-Service (SaaS) has increased significantly in the past few years. One use of SaaS in High-Performance Computing (HPC) is in the form of "software licenses." Software licenses give temporary access to commercial software for the jobs/applications scheduled to execute in large HPC clusters. Normally, a job scheduled for execution in such clusters cannot start its useful work without proper corresponding software license(s) even if it has already acquired its necessary hardware resources. Therefore, for many types of HPC workloads, license management is, in principle, as important as hardware resource management.

Unfortunately, compared to hardware resource management, which has enjoyed substantial investigation [15], [19], [22], [25], license management has not received much attention from the research community. This is partly because most HPC application/hardware owners tend to heavily *over-provision* licenses (purchasing more licenses than required), to avoid potential scenarios where a job acquires the hardware resources it needs for execution but could not acquire the required software licenses. However, the cost of such over-provisioning has recently reached intolerable levels, with large companies spending up to 18% of their revenue on software [10].

Ideally, only the "right" number of licenses would be purchased, and the number of licenses provisioned would vary along with time-dependent needs, thereby reducing the overheads of licenses purchased to support peak demands that may never or rarely occur. However, being able to do so requires accurate *prediction* of *long-term* license needs. While there have been a few preliminary works that have focused on this problem, the findings so far are not satisfactory and, as a result, the proposed approaches have not, to our knowledge, been deployed in any production system. Interestingly, in addition to long-term prediction, which can guide license provisioning/purchase decisions, *short-term* license usage prediction can also be very useful in practice. In particular, short-term prediction can provide additional input to a *job scheduler*, enabling it to make, for example, "license availability-aware" scheduling decisions. In other words, **predicting the number of licenses that would be needed in the future, whether that is near-term or long-term, can benefit both the managers and users of HPC clusters**.

This work focuses on performing long *and* short-term license prediction (*forecasting*). By performing long-term (months-year) forecasting, HPC administrators can make informed decisions when purchasing licenses, to save money. Short-term prediction, on the other hand, can be used to improve job scheduling, allowing users of a cluster to execute their jobs sooner and have less work wasted due to license-related job failures, allowing HPC administrators to use fewer resources while still satisfying service-level agreements.

The main **contributions** of this paper include:

- We introduce a method to process HPC log/accounting files, for use in forecasting and for simulation.
- We evaluate several forecasting models for long-term forecasting, show the results for the best model and propose a method for short-term forecasting.
- We propose a "license-aware" job scheduling strategy that uses short-term license usage forecasts.
- We design a simulator that can simulate job and license behavior in regard to changes in the job scheduler and license manager. We evaluate our scheduler on this simulator and our experimental analysis of the proposed license prediction, compared to the baseline method of "polling", indicates that for periods of high license usage: (i) the number of denied jobs due to insufficient licenses decreases by up to **96%**; (ii) as a result, the amount of wasted work decreases by up to **92%** with close to no change in job completion times.

## II. Background

This section provides an overview of current usage for software licenses (SaaS) and license managers. It then presents background on our chosen forecasting model – A Multi-Horizon Quantile Recurrent Forecaster (MQ-CNN).

Software licenses are contract agreements between software publishers and end-users of an application [7]. Licenses protect the software vendor's rights in cases of duplication, multi-host installation, code editing (unless the product is open-source), and reverse engineering [7], [8]. Software licenses are often structured as collections of several different sub-licenses which unlock different functionalities of the software. For example, a MATLAB license consists of features such as `Symbolic_Toolbox`, `Statistics_Toolbox`, `Neural_Network_Toolbox`, `Bioinformatics_Toolbox`, and more.

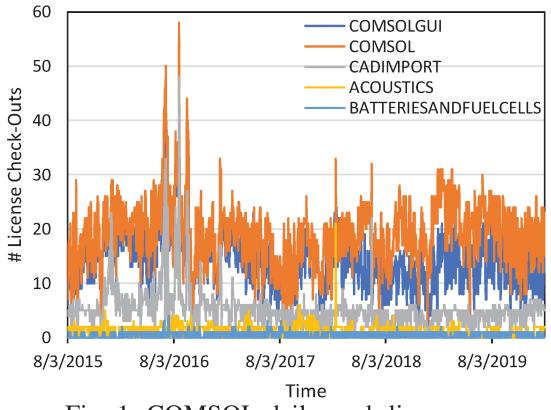


Fig. 1: COMSOL daily peak license usage plot.

Software publishers/vendors use license management mainly for protection from piracy and protection against exceeding license terms. License management also allows the publishers to create different types of licenses (trial/perpetual/subscription) and set usage limits for individual license features or the whole license [7]. Existing licensing models differ in terms of the number of licenses purchased to the resources available to install licensing systems, but fall, broadly, in a modest number of categories [8].

In this study, our focus is on the *FlexNet license manager (FlexLM)* [9]. The FlexNet License Manager, formerly known as FlexLM, is a popular license management product that allows the management and control over the distribution of software licenses for various software applications. FlexNet allows software vendors to implement various licensing models, such as Node Locked licenses and Floating (concurrent) licenses [9]. The details of FlexNet and other commonly-used license managers can be found elsewhere [7], [9].

In this work, we collect historical data on license usage. This collected data is time-series data, and there are a number of preexisting approaches to forecast from time-series data in the literature. We used various supervised learning models to perform forecasting, and we only list the results of the best-performing model, which is MQ-CNN. MQ-CNN is a Seq2Seq[1] framework that solves complex forecasting problems and generates Multi-Horizon Quantile forecasts [27].

## III. Motivation

Many HPC applications require specialized commercial software, which can present significant costs to businesses and application owners alike. In HPC environments, commercial software generally requires licenses, which are purchased for a limited period of time, to utilize them. Most companies and institutions *over-provision* software licenses to avoid facing the scenario of application failures due to license shortage. Consequently, license costs can be substantial, and accurate long-term forecasting of license usage is crucial for avoiding over-provisioning these licenses and thus minimizing the associated (unnecessary) expenses made by businesses using license-over-provisioned HPC environments.

[1]A Seq2Seq model takes a sequence of items and outputs another sequence of items [26].

Moreover, the use of software licenses is expected to grow significantly in the coming years, further increasing the costs of over-provisioning licenses. In 2022, the software global market share had a value of $51.08 billion, with over half of this value from software subscriptions alone, and it is estimated to reach a market value of $91.6 billion in 2028 [6]. In a 2022 survey of 501 organizations with 2000 or more employees, with half having over 10,000 employees, it is revealed that the median fraction of the revenue that is spent on software is about 18% [10].

In addition to cost savings, forecasting license usage can help incorporate "license awareness" into job schedulers and resource managers. Many HPC systems have separate software for managing HPC hardware resources and software licenses. However, these two groups of software work independently of one another, which is unfortunate because for a job to start and successfully execute *both* hardware resources and software licenses must be acquired. By predicting whether a job would fail due to insufficient licenses using short-term forecasting, job schedulers can be made more license-aware. While in this study we collect data from The Pennsylvania State University (PSU) ROAR HPC cluster to make forecasts based on the collected data, we believe that the problem of "disconnect" between hardware resource management and software licenses is endemic across different institutions worldwide [11]. We believe our cluster represents a typical mid-size datacenter targeting HPC workloads that need widely-used software licenses.

### A. Long-Term Forecasting

For businesses, purchasing licenses can cost a significant fraction of their revenue [10], making even a small decrease in the number of licenses purchased significant. For example, for COMSOL Multiphysics' 2015 pricing, a single base COMSOL floating network license, which can be used by a single user concurrently, costs $3,390 annually, with additional sublicenses costing between $590 to $3,390 [17]. As mentioned previously, licenses can be underutilized by more than 30%. With accurate long-term forecasting of license usage, businesses and HPC managers can make informed decisions and mitigate the over-purchasing of software licenses. Using COMSOL software license logs obtained from PSU ROAR HPC, we observe that licenses are very sparsely used. The base COMSOL license on average has 12.57 licenses checked out at a given point in time, yet the total number of base COMSOL licenses purchased are 51, giving a utilization rate of 24.6%. This stems from licenses being purchased to accommodate "peak" usage, *not* "average" usage. By taking the peak usage license usage per day for the base COMSOL license and averaging it, we obtain a daily peak average of 19.14 and a standard deviation of 8.45, giving a daily peak utilization of 37.2%. Analysis of our COMSOL license data shows that the highest daily peak utilization in our data ranges from 0.06% to 70.2%. Fig. 1 shows the daily peak license usage for our logs. Note that, while there are a total of 49 different COMSOL licenses in our log data, for ease of visualization, this plot only

shows the base COMSOL license (*COMSOL* in the figure) and the four "most-denied" licenses.

### B. Short-Term Forecasting

Job schedulers and license managers are usually two distinct pieces of software in HPC environments. In PSU's ROAR HPC cluster, the job scheduler (TORQUE) has *no* information about licenses/license management, and the license manager (FlexNet LM) is *not* aware of scheduling decisions. Licenses can be requested at any point during job execution, which is completely unknown to the job scheduler. By utilizing short-term forecast information of future license usage/requirements, one can predict whether a job will fail due to insufficient licenses and schedule the job when it has a lower chance of failing. This helps to prevent the potential loss of work and time due to unpredictable jobs failing.

Furthermore, in the ROAR HPC cluster, jobs that fail to check out a license due to insufficient available licenses, simply get *terminated*. Unless the user running the job had saved the job progress, the resubmitted job needs to *start over* and re-perform any work that was done before. This leads to a waste of hardware resources and energy, and it negatively impacts users. Additionally, in periods of high amounts of license denials, empirically, users typically resubmit jobs until their job manages to check out the licenses it needs. This leads to users resubmitting the same jobs hundreds of times in a short period, causing additional stress on the job scheduler and license manager, and *decreases the job's priority* due to fairshare policies in the job scheduler [5]. From our 4-year COMSOL license data set, for example, we observe that 43,421 job failures are directly caused by a lack of licenses, and these failed jobs were submitted by 555 distinct users.

### C. Simulation

In our work, we implemented an accurate *simulator* that can use historical data to simulate a past time period. This can be used to estimate how the changes made to the job scheduler, licenses, and license manager affect jobs, without having to test them in a real HPC environment. The simulator also enables us to test the effectiveness of novel/emerging license-aware job scheduling strategies which may be infeasible to implement in practical job schedulers used today.

## IV. Design

This section explains the log processing methods, the design of the forecasting model, and the simulator design.[2]

### A. Preprocessing

*1) Processing License Logs:* For our experiments, we used log data obtained from PSU's ROAR HPC system, for the COMSOL [4] software ranging from August 2015 to February 2020. COMSOL Multiphysics is a software platform that is based on advanced numerical methods. It is used for physics-based problem modeling and simulations [4]. In our setup, the COMSOL log files are generated from the FlexNet LM software [9]. In total, there are 4,568,089 intervals of license check-outs and check-ins we obtained from this log. In this context, a "check-out" means that a license was *borrowed* by a job, a "check-in" means that a license was *returned* by the job, and denied means that a license was attempted to be checked-out, out but failed due to some reason (such as no currently free license). In addition, our data contain a total of 49 licenses, one of which is the base COMSOL license, which unlocks basic features of the COMSOL software, and the 48 other sub-licenses that unlock additional functionalities.

[2]Code & data: https://github.com/abgulhan/LicenseManagementMASCOTS

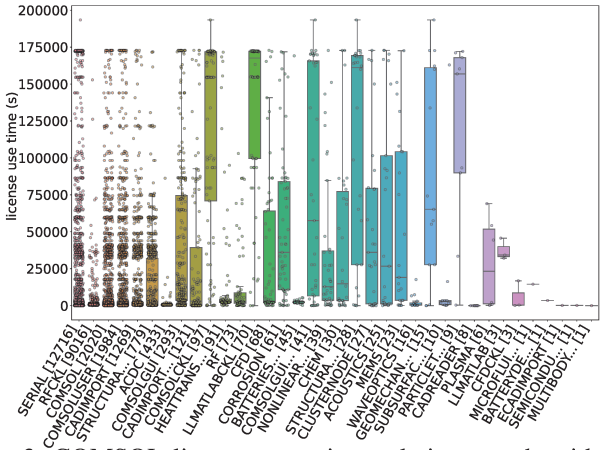



Fig. 2: COMSOL license usage intervals in seconds with the number of data points in brackets for a 2 month period.

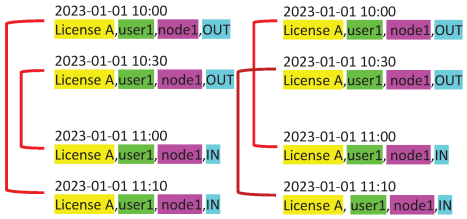



Fig. 3: License matching problem illustration.

In the FlexNet License Manager, relevant lines in the log file contain the date-time, the action taken (whether a license was checked-out, checked-in, or denied), name of the license for which the action was attempted, ID of the user attempting the action, and compute node in which the action was taken. The time granularity is in *seconds*. The FlexNet log file, in its current form, *cannot* be used for simulation and forecasting, as it needs to be processed into suitable "layouts" for each. To process the log file, we first match license check-outs and check-ins into a single interval. A username, compute node, and feature name must be the same between a check-out and its corresponding check-in, as shown in Fig. 3. However, while this does match most licenses without issue, there is still ambiguity for 8.56% of all licenses. One reason for this ambiguity is that a user may check out duplicate licenses from the same compute node. This issue is illustrated in Fig. 3. To deal with this issue, we decided to choose pairs that minimize license use duration, since jobs in our system have a wall-time limit of 48 hours, by choosing the minimum duration, we can be sure that we satisfy this condition, thus eliminating some incorrect possibilities. Fig. 2 shows a plot of the license intervals for 2 months of log data. We can observe that almost all license usage intervals are below 2 days. Note that this assumption does not change the number of licenses currently

being used in any time period: It only affects the job intervals which are used in the simulation. Another reason is that there are some missing data in the log files from when the license manager was restarted. To deal with this issue, we identify the regions where the license manager was not logging data, which are usually only a few hours in duration. Then, for check-outs that do not have a valid check-in and vice-versa, we assign a check-in or check-out from these regions.

This license matching problem can be formulated and solved using "minimum weight bipartite matching" [13]. We make a bipartite graph where sets U and V represent the two disjoint independent sets. Each node in set U represents a check-out, and each node in set V represents a check-in. Edges between U and V are made if the check-in node in V has a larger date-time than the check-out node in U and if the difference between these two is smaller than the maximum job length. The edge weight corresponds to the number of seconds between the date-times of the two connected nodes. Edges chosen after solving minimum weight bipartite matching are the license intervals that are matched. Nodes disconnected from the chosen edges are candidates for having missing data in regions where the license manager stopped logging data.

*2) Processing Accounting Logs:* The accounting log contains information about submissions to HPC job queues and is produced by TORQUE [24], which is based on the Portable Batch System (PBS) job scheduler [3]. Our accounting log ranges from October 2016 to January 2023 and contains over 120 million data points for 40,634,793 different jobs. However, only a fraction of these jobs use COMSOL software licenses. This log does *not* contain any license information. It only contains job-related information, of which there are three relevant types of data points indicating: (i) when a job was queued, (ii) started execution, and (iii) ended execution. These data points all contain a unique job ID, the corresponding time, and the user who submitted the job. Data points (ii) and (iii) also contain the compute node(s) where the job is executing. We use accounting logs to simulate jobs for testing our job scheduler. However, just the accounting log is not sufficient for this purpose since jobs can use licenses at various periods throughout their execution, which can affect if a job finishes successfully or terminates due to insufficient licenses. Therefore, we need to *match* the license intervals obtained in Section IV-A1 to their respective jobs, (explained in Section IV-A3).

*3) Matching Licenses to Jobs:* When matching licenses to jobs, our goal is to find a job ID for each license usage interval. Doing so allows us to know when, during a job execution, a license was requested and how long it was used. Note that multiple license usage intervals can map to the same job. Since the license logs do not contain job ID information, we instead use the user name, compute node, and time stamp information, which appears in both accounting and license logs. Correct matching of a license usage interval to a job must satisfy the following conditions: (i) the compute node where the license was requested and the compute node a job is running on must match; (ii) the user who submitted the job and the user who requested the license must match; and (iii) the job execution start time must be less than or equal to the license check-out time and the job execution end time must be larger than or equal to the license check-in time. However, even if these conditions are satisfied, there could still be some ambiguous matching. Several jobs may satisfy the above conditions, since multiple jobs may be run by the same user on the same compute node. An example is shown in Fig. 4. The license interval shown in both figures satisfies the conditions stated above, yet it is still not clear which job this license interval belongs to. This job mismatch occurs in 68% of jobs using COMSOL licenses in our data and the average number of possible jobs per mismatched license is 2.48. Furthermore, license usage intervals themselves may also not be correct, since there are ambiguous matches between license check-ins and check-outs, as mentioned earlier in Section IV-A1. Since matching occurs in just 2 jobs per license interval in a majority of cases, and since no correct solution for this exists, due to it being caused by insufficient information, we randomly match jobs to such licenses upon such ambiguity.

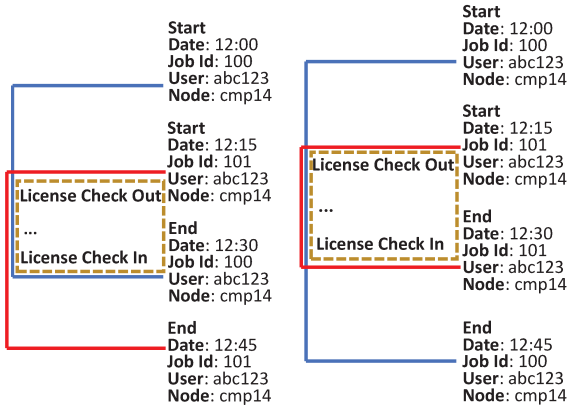


Fig. 4: Ambiguity examples for matching licenses to jobs.

## B. Generating Forecasting Data

For our forecasting model, we used the "license usage interval" data we obtained, as described in Section IV-A1. However, we need to *transform* this data into a layout suitable for long-term forecasting. To do this, we used a "wide" data representation, where the index values are date-time, and the other columns represent the total number of licenses checked out at a given date-time index. Since we have a total of 49 COMSOL licenses in our license log, the data set has 49 columns, each representing a different license plus one index column. The date-time values for indexes correspond to license intervals' check-in or check-out, since only at those times does the total number of licenses in use change.

The wide data representation, as is, cannot be used for forecasting, since the index values need to be equally spaced. For long-term license forecasting, we chose 24-hour spacing, meaning that the forecasts are done with a 24-hour granularity. The reason for this is that the more data points that are being forecasted, generally the less accurate the forecast is and the slower the model training is. As an example, with

the DeepAR forecasting model, forecasting over 400 data points is not recommended for this reason [2]. With long-term forecasting, we perform up to 365 days of forecasts, and with 24-hour spacing, we stay under the recommended maximum data points. Having a smaller spacing is also unnecessary since the goal of long-term forecasting is to predict the license usage trend, for making informed decisions when purchasing licenses. Furthermore, when evenly spacing the wide data, we chose the "maximum number of licenses used", for each license, in the chosen even-spacing interval. We choose the maximum, since we want to predict peak license usage, not the average, to prevent license denials. Further, license usage is sparse and bursty – the peak and average are quite far apart for most licenses observed.

### C. Long-Term Forecasting Model

We tested several forecasting models, including, ARIMA [14], DeepAR [21], DeepVAR [20] and MQ-CNN [27], and found that MQ-CNN performs the best for our data. MQ-CNN is a Seq2seq-based CNN forecasting model and is used to make multivariate forecasts, meaning that the features that it is trained on do not need to be known in the future forecasting period. This allows MQ-CNN to make use of multiple license information. We use the GluonTS [1] library's implementation of long-term forecasting models.

### D. Short-Term Forecasting Model

For short-term forecasting, we take a different approach. The purpose of short-term forecasting is to be used during job scheduling. Whenever a job fails due to an insufficient license, we want to *delay* that job upon resubmission until that license becomes free. The goal is to predict the earliest time a license becomes free, whereas in long-term forecasting the goal was to predict how many licenses are in use at each time period. While the long-term forecasting approach can be used to figure out when a license becomes free, we found that the forecasting models were *not* accurate enough for this purpose. Instead, we propose a method of predicting "when" a license becomes free by using a license usage distribution that is derived from historical license usage times – which requires license log files or another method of recording license use times. The time for delaying a license, that is how long to wait until a license becomes free, is formulated as an expected value as follows:

$$\begin{aligned} Delay = E(min(X_1 - t_1, X_2 - t_2, ..., X_n - t_n)| \\ X_1 > t_1, X_2 > t_2, ..., X_n > t_n), where X_i \sim f(x), \end{aligned} \tag{1}$$

where $X_i$ is a random variable, denoting the license usage duration of a currently in-use license $X$, which has been running for time $t_i$, where license usage times are from a probability density function $f(x)$. This equation can also be formulated as follows:

$$\int_0^\infty \frac{y \frac{d}{dy}(1 - \prod_{i=1}^n Pr(X_i > y + t_i))}{\prod_{i=1}^n Pr(X_i > t_i)} dy \tag{2}$$

We implemented (2) numerically where the distribution for $X_i$ was obtained using license interval data from IV-A1 and is updated as new license intervals are completed during simulation. To deal with numerical stability issues that arise from having a large $n$ value, that is, having many of that license currently in use when predicting the expected delay for a license, we do not calculate $Pr(X_i > t_i)$ and $Pr(X_i > y + t_i)$ directly. We instead omit the denominator of these probabilities so that we obtain an integer instead of a probability between 0 and 1. Since this omitted value occurs in *both* the numerator and denominator of the *integral* in (2), it does *not* affect the result, but it significantly increases numerical stability.

It should be noted that this is a limited form of forecasting license usage, compared to our long-term forecasting method. Our short-term forecasting method only predicts when the license usage will **decrease**; it does not predict license increases. Furthermore, it assumes that the predicted license will not increase past its current value, before decreasing. Since this forecasting method is only used when a license is at its maximum usage, the aforementioned assumption is satisfied and the limitation is sufficient for job scheduling purposes. Note that, this method, in its current form, *cannot* be used to predict the case where *multiple* of the *same license* are checked-out simultaneously by the *same job*. Our COMSOL dataset *does* have such licenses, but all of those licenses are *unlimited*, therefore those licenses are never denied.

### E. Simulation

*1) Assumptions:* Due to limitations in extracting information from log files, we make several assumptions, listed below. It should be noted that, in real-world use cases, these assumptions may not hold, which may lead to different results:

- **Job Independence**: We assume that each job is submitted independently of other jobs, but this is not always the case. A user may submit a subsequent job only after a previous job completes, or users may use workflow managers, that splits a task into multiple dependent jobs. However, since such information is not found in our accounting logs, we make this assumption.
- **Stateless Jobs**: We assume that once a job fails, the resubmitted job will take the same amount of time to complete and use same licenses at the same periods of execution. This assumption will only hold if the resubmitted job does not use work done by its previous failed execution. For example, a user may make checkpoints during job execution and resume from the most recent one after a job failure, leading to resubmitted job taking less time to complete and potentially using different licenses for different periods of time. Since this information is also not available in our log files, we make this assumption.
- **Job Resubmission**. In our job submission system, when a job fails, it needs to be *resubmitted* by the user. This can be done by the user manually resubmitting, or writing a script to automatically resubmit. There may be a few seconds of delay between when a user submits a job and when a job is logged in the accounting log, and similarly when a job fails and when the user receives the error message. Furthermore, we cannot know when a user will decide to resubmit a job, and how often they will continue trying to resubmit. For these reasons, we need to make an assumption about the delay between job re-submissions in our simulation. We can observe

the resubmission behavior in our log files, and see that the it *varies*. We use the "average" of resubmission intervals of log data in our simulated time period as the "resubmission time" for jobs and we continue resubmission until success.

• **Queuing time**. Job queuing time can vary significantly depending the requested hardware resources. Due to the complexity of accurately simulating these compute resources, we make the assumption that jobs will have the "same" queuing time recorded in the accounting logs upon resubmission.

### F. Simulator Design

Our simulator, shown in Fig. 5, is used to simulate the behavior of historical data and test the changes we make to the job scheduler. The input data to the simulator consists of jobs, with each job having:

• *submission time*, which is equivalent to the job queue time we extracted from the accounting logs in Section IV-A2;

• *duration*, which is the time between job start and job end, also discussed in Section IV-A2;

• *queue name*, which indicates which queue the job will be submitted to. This information is from accounting logs;

• *license usage intervals* of that job, if any. As shown in Fig. 5, a job can have licenses used at any part of its execution. Licenses each have a check-out time and the license usage duration, at the end of which the license is checked-in. This matching was explained in Section IV-A3. Not all jobs have licenses, yet those jobs can optionally be used in the simulation if simulating limited nodes is desired, since those jobs *affect* the queuing time of other (license-using) jobs.

Jobs are read from the input file and sent into a job queue when job submission time arrives. The job queue consists of a high-priority "batch" queue, which is reserved for users who paid for resource allocation, and a low-priority "open" queue, which can be used by *all* users for free. Any jobs in the high-priority batch queue begin execution before the open queue. Jobs begin execution when they are assigned to a compute node. Each compute node requests a license from "license manager" when/if a job checks out a license at the current time. When a license is finished being used by the job, the license is checked into the license manager. The license manager contains several licenses along with their respective maximum amount. Upon check-out, one of the licenses is assigned to the job requesting it and becomes unavailable. If a job requests a license that is not available at that time, the job terminates and is sent to the "resubmission queue", as shown in Fig. 5. The resubmission queue puts jobs back into the job queue after a certain "delay". This delay can be either chosen as a fixed/constant value, which can be calculated using the accounting logs (to attempt to simulate close to historical data), or it can be chosen based on a short-term forecasting model. The forecasting model *predicts* when the license that the job failed to check out would become available and sets a delay value accordingly. The resubmitted job, starts execution from the beginning, *not* from where it got terminated.

## V. Experimental Setup

We used Python 3.7.12, Pandas 1.2.5, Apache MXNet CUDA 1.6.0, GluonTS 0.12.5, and NumPy 1.21.6, SciPy 1.7.3

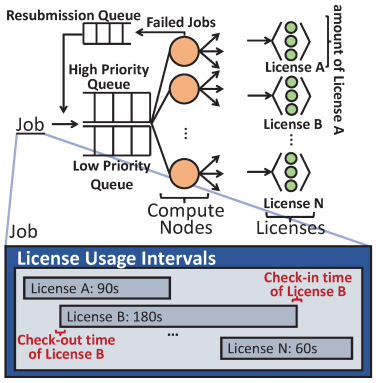



Fig. 5: Simulator design.

| License Name | COMSOL | COMSOLGUI | CADIMPORT | ACOUSTICS |
|---|---|---|---|---|
| MSE | 39.574 ± 3.178 | 18.406 ± 0.32 | 3.886 ± 1.66 | 4.29 ± 0.895 |
| MAPE | 0.294 ± 0.003 | 0.307 ± 0.04 | 0.439 ± 0.0 | 0.625 ± 0.045 |

TABLE I: 1-Year Forecast Results

library versions for the forecasting model, simulation, and data processing stages.

The license logs that are used as input to our forecasting model are collected with FlexNet Licensing version v11.15.1.0. The software platform for which the licensing logs are collected is COMSOL Multiphysics, which has been described earlier. The PSU ROAR HPC infrastructure operates 30,000 cores to support research and provides dual 10- or 12-core Xeon E5-2680, and 10-core Xeon E7-4830 processors, with memory configurations of 128GB, 256GB, and 1TB respectively. The operating system is RHEL 7.

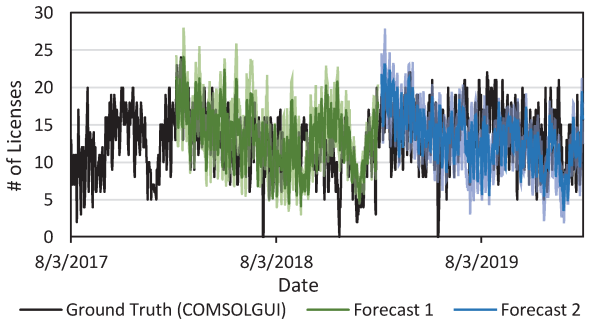
## VI. Experimental Results

### A. Long-Term Forecasting

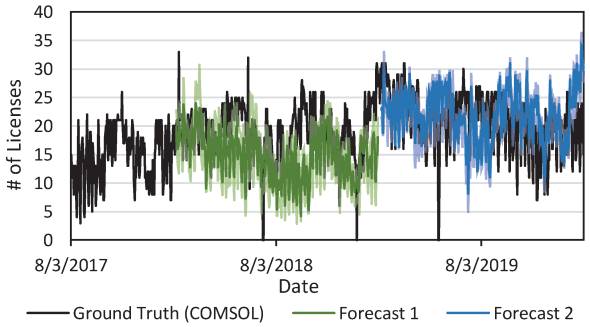
For our long-term forecasting results, the best-performing model that we tried was **MQ-CNN**. We trained our model using 2 years of data and forecasted 1 year in the future. Fig. 6 shows our best 1-year forecast results for licenses COMSOL and COMSOLGUI – two of the most commonly used licenses. This figure shows two separate forecasts, in green and blue, where each forecast was trained independently of the past 2 years of data and predicted the following year. Since MQ-CNN is a "quantile-based" forecasting method, we trained the model for 3 different quantiles of 0.05, 0.5, and 0.95. The Mean Average Percentile Error (MAPE) and Mean Squared Error (MSE) results are listed in Table I. For each element in the table, we average the resulting metrics of both forecasts and give the standard deviation. We observe that licenses with low maximum amount, such as CADIMPORT and ACOUSTICS, have higher MAPE, but lower MSE.

### B. Job Scheduling

We focus on predicting a suitable resubmission delay for jobs that were denied due to insufficient licenses, such that, after the delay, the necessary licenses would be available. For our simulation data, we chose a total of 3 months of simulation data with the highest number of license denials that were *not* caused by interactive nodes, since interactive nodes do not simply fail on license denial. To measure the performance of




(a)




(b)

Fig. 6: 1-Year Forecasting results of maximum licenses used per day using the MQ-CNN Model, with quantiles of 0.05, 0.5, and 0.95. (a) shows the forecasting results vs baseline for the COMSOLGUI license and (b) shows the forecasting results vs baseline for the COMSOL license. Green and Blue lines each show a 1-year forecast, using the MQ-CNN model which was trained with the previous 2 years' data.

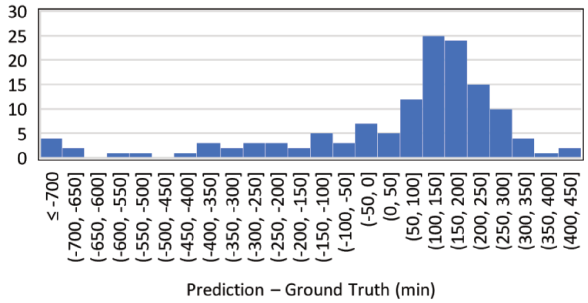
| Forecast Type (Delay) | # Job Denials | # Unique Job Denials | Avg. License-Using Job Time | Avg. Denied Job Time | Total Wasted Work | Avg Wasted Work Per Denied Job |
|---|---|---|---|---|---|---|
| Constant | 3991 | 39 | 9:23:59 | **1 d, 2:15:32** | 2 d 7:17:13 | 1:25:13 |
| Exponential | 166 | **22** | 9:58:00 | 5 d 8:59:12 | **4:07:57** | 0:11:16 |
| Forecast | **135** | 32 | **9:22:51** | 1 d 4:27:48 | 4:11:18 | **0:07:51** |

TABLE II: Job scheduling results.

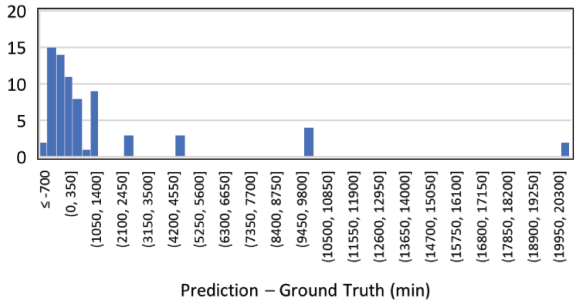
our job scheduler, for our evaluation metric, we use the amount of wasted work due to job denials and the average time from job submission to job completion. We compare three different scheduling methods:

- **Constant Denial Time:** This method uses the historical average job resubmission delay for every job resubmission.
- **Exponentially Increasing Denial Time:** This method starts with the historical average job resubmission delay for the first job resubmission and then doubles this time upon every subsequent resubmission of the same job.
- **Forecasted Denial Time:** This method calculates a delay using historical license usage distribution information, as described in Section IV-D.

From simulating multiple periods of high denial time, (Table II), we observe that the least number of job denials were obtained by using the forecasting delay method, which has a 92% decrease in the amount of wasted CPU time and a small decrease in job completion time – the time from a job's first submission to its completion. Exponential delay has the least wasted work, at the cost of significantly increased job completion time. Using constant delay has the highest job




(a)




(b)

Fig. 7: Histogram of distribution-based forecasting accuracies during simulation for the forecasting (a) and exponential delay (b) methods. The x-axis represents the difference between forecast and ground truth in minutes.

denials and average wasted CPU time per denied job, but it has the lowest average denied license-using job completion time, resulting from the polling behavior of this method leading to almost no overprediction, at the cost of wasting hardware resources and increased denials. In our simulation, we assume job queue time is unaffected by currently running jobs, as explained in Sec. IV-E. However, in reality, the total wasted work and denials *would* negatively affect the queue time of jobs, due to occupying hardware resources and fairshare policies, leading to higher job completion times than those reported in our simulation results.

We analyzed how accurately our job scheduler predicts when a license will become free by recording each prediction and ground truth during our simulation. The results are shown in Fig. 7 for the exponential and forecasting method. For the forecasting method in Fig. 7a, we see that the model clusters around the prediction and ground truth difference of 0 minutes, with more predictions on the positive side, indicating that the model can predict accurately, but tends to overpredict. Note that underpredictions lead to job denials and further delay predictions. For the exponential method in Fig. 7b, we see that there are many underpredictions with few very large overpredictions, which causes increased job completion time.

## VII. Related Works

In the context of *license-aware job scheduling*, in [12], the authors propose a license-aware job scheduler by adding a queue on top of an existing job scheduler and only running jobs when both the license manager and the job scheduler indicate that resources are available. However, this depends on the assumption that what licenses a job will use is known

before execution and it results in idling on hardware resources. In [29], a heuristic is presented and evaluated for license-aware job scheduling. However, the proposed heuristic assumes that the licenses used in a job and the runtime of a job are known *before* execution. We believe, due to these assumptions, such methods are *not* suitable for many HPC environments. In contrast, our work can be *integrated* with existing job schedulers without requiring any future knowledge.

In the context of *license usage forecasting*, there exist commercial tools, such as X-formation License Statistics [28] and OpeniT License Predictor [16], which claim to work with several types of software licenses. However, these solutions are *not* open-source. Few works exist on license forecasting such as [23] which uses the ARIMA forecasting algorithm [14] to predict petroleum software and [18], which compares an LSTM-based forecasting model with the OpeniT License Predictor. However, these works contain insufficient details of their implementations and are of limited use. In contrast, we give the details of our proposed methods, and our software will be put in the public domain.

## VIII. Concluding Remarks and Future Work

Our work shows that it is possible to accurately forecast license usage, for up to a year, and we show that with the ability to accurately forecast license usage in the short term, it is possible to improve job scheduling in a license-aware manner. We also show methods of parsing log files to obtain data for use in forecasting. We plan to improve the simulator by tracking CPU use, both cores and memory per CPU, to simulate queue time instead of using historical queue time in the future. Furthermore, we plan on adding compatibility for interactive jobs both by identifying them in the log data and simulating them correctly. We plan to analyze how forecasting errors impact job performance and investigate different license and job scheduling environments, including the requirement of additional information at job submission, e.g. license and workflow annotations. For the distribution-based forecasting model, we plan to add a decaying distribution, so it can adapt to recent changes in license durations and we plan on updating this model to be able to deal with multi-license checkouts.

## Acknowledgment

The data used in this research are collected on the Pennsylvania State University's Institute for Computational and Data Sciences' ROAR supercomputer. This research is supported in part by NSF grants 2122155, 1822923, 1931531, 2211018 and 1908793, and Department of Energy grant DE-SC0023186.

## References

[1] A. Alexandrov *et al.*, "GluonTS: Probabilistic and Neural Time Series Modeling in Python," *Journal of Machine Learning Research*, vol. 21, no. 116, pp. 1–6, 2020. [Online]. Available: http://jmlr.org/papers/v21/19-820.html

[2] Amazon Web Services, Inc, "Amazon sagemaker developer guide use built-in algorithms: Deepar forecasting algorithm," Amazon Web Services, Inc, Tech. Rep., 2023.

[3] A. Bayucan *et al.*, "PBS Portable Batch System," MRJ Technology Solutions, Tech. Rep., 1999.

[4] COMSOL, "COMSOL license options for commercial and academic use," 2023. [Online]. Available: https://www.comsol.com/products/licensing

[5] T. M. Declerck and I. Sakrejda, "External torque/moab on an xc30 and fairshare," *Technical report, NERSC*, 2013.

[6] N. G. et al., "Market share: All software markets worldwide 2022," Gartner, Tech. Rep., 2023. [Online]. Available: https://www.gartner.com/en/documents/4273199

[7] D. Ferrante, "Software licensing models: What's out there?" *IT Professional*, vol. 8, no. 6, pp. 24–29, 2006.

[8] D. Ferrante, "Software licensing models: What's out there?" *IT Professional*, vol. 8, no. 6, pp. 24–29, 2006.

[9] Flexera, "Flexnet manager," 2023. [Online]. Available: https://www.flexera.com/products/flexnet-manager

[10] Flexera Software, "Flexera 2022 tech spend pulse," Jun 2022. [Online]. Available: https://www.flexera.com/about-us/press-center/flexera-2022-tech-spend-pulse-provides-insights-into-enterprise-it-spend

[11] Flexera Software, "Flexera 2023 State of the Cloud Report," Mar 2023. [Online]. Available: https://info.flexera.com/CM-REPORT-State-of-the-Cloud-2023-Thanks

[12] Z. Hou *et al.*, "A software licenses aware job scheduling and management approach on multi-clusters," in *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*. Paris, France: IEEE, 2016, pp. 706–711.

[13] R. Jonker *et al.*, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, pp. 325–340, 1987.

[14] B. K. Nelson, "Time series analysis using autoregressive integrated moving average (arima) models," *Academic Emergency Medicine*, vol. 5, no. 7, pp. 739–744, 1998.

[15] V. Nollet, T. Marescaux, P. Avasare, D. Verkest, and J.-Y. Mignolet, "Centralized run-time resource management in a network-on-chip containing reconfigurable hardware tiles," in *Design, Automation and Test in Europe*, IEEE. Munich, Germany: IEEE, 2005, pp. 234–239.

[16] Open IT, "Open IT License Predictor," Jan 2023. [Online]. Available: https://openit.com/products/licensepredictor/

[17] Purdue University, "Comsol Licensing," May 2021. [Online]. Available: https://engineering.purdue.edu/ECN/Support/KB/Docs/ComsolLicensing

[18] T. Rosenwinkel and L. Cole, "Forecasting software license usage using machine learning in a predictive analytics platform," in *NAFEMS World Congress*. Quebec, Canada: NAFEMS Ltd, 2019.

[19] A. Rowstron and P. Druschel, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '01. New York, NY, USA: ACM, 2001, p. 188–201.

[20] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus, "High-dimensional multivariate forecasting with low-rank gaussian copula processes," 2019.

[21] D. Salinas *et al.*, "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks," 2019.

[22] E. Šlapak *et al.*, "Cost-effective resource allocation for multitier mobile edge computing in 5g mobile networks," *IEEE access*, vol. 9, pp. 28 658–28 672, 2021.

[23] M. Song *et al.*, "Petroleum software license usage forecast based on arima algorithm," in *2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC)*. Guiyang, China: IEEE, 2021, pp. 336–340.

[24] G. Staples, "Torque resource manager," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, ser. SC '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 8–es.

[25] M. Stillwell *et al.*, "Resource allocation algorithms for virtualized service hosting platforms," *Journal of Parallel and distributed Computing*, vol. 70, no. 9, pp. 962–974, 2010.

[26] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

[27] R. Wen *et al.*, "A multi-horizon quantile recurrent forecaster," in *NeurIPS 2017*. Long Beach, CA: NeurIPS, 2017.

[28] X-Formation, "License statistics – software asset management tool," Oct 2021. [Online]. Available: https://www.x-formation.com/license-statistics/

[29] D. Xiaoshe *et al.*, "A license-aware meta-scheduling engine in grid environment," in *The Third ChinaGrid Annual Conference (chinagrid 2008)*. Dunhuang, China: IEEE, 2008, pp. 55–61.