Abstract. The increasing volume of physics data poses a critical challenge to the ATLAS experiment. In anticipation of high luminosity physics, automation of everyday data management tasks has become necessary. Previously many of these tasks required human decision-making and operation. Recent advances in hardware and software have made it possible to entrust more complicated duties to automated systems using models trained by machine learning algorithms. In this contribution we show results from one of our ongoing automation efforts that focuses on network metrics. First, we describe our machine learning framework built atop the ATLAS Analytics Platform. This framework can automatically extract and aggregate data, train models with various machine learning algorithms, and eventually score the resulting models and parameters. Second, we use these models to forecast metrics relevant for network-aware job scheduling and data brokering. We show the characteristics of the data and evaluate the forecasting accuracy of our models.

1. Introduction
The data taken by the ATLAS Experiment [1] at the Large Hadron Collider (LHC) are stored and distributed via the Worldwide LHC Computing Grid (WLCG), a network of computing centres and users across the globe. Due to the heterogeneous nature of the WLCG, the transfer model depends on a complex amalgamation of different systems, each with different queuing, execution, fail-over, and retrial strategies [2]. This can lead to wildly varying data transfer times and eventually to infrastructure under-utilisation and user dissatisfaction. One of the potential solutions is to place experiment data depending on their potential usage instead of just using the fixed distribution percentages from their associated computational workflows. This should lead to fewer transfers and higher overall system throughput. Such new strategies require the usage of infrastructure reliability and performance metrics in the data placement algorithm in a dynamic feedback cycle. Using these extra metrics presents unique challenges and this article describes the state of our ongoing work in this area.

To quantify any improvements in data placement we need an estimator of potential transfer times. A global minimum solution for all expected transfers yields, in theory, an optimum placement strategy when assuming equivalence of single computational resources. Geographical distance or network bandwidth are insufficient quantifiers though. The dynamic nature of both the involved applications and the networks themselves, as well as their physical deployment, require a more sophisticated method. Currently there is no sensible model available to estimate the duration of a particular collection of file transfers between data centres.
Table 1. Available DDM network metrics.

| Key          | Aggregation       | Interval     | Source   | Interface      |
|--------------|-------------------|--------------|----------|----------------|
| src          | -                 | 1 hour       | AGIS     | AGIS           |
| dst          | -                 | 1 hour       | AGIS     | AGIS           |
| closeness    | -                 | 1 hour       | AGIS     | AGIS           |
| packetloss   | 95th pct. last hour | 10 min. perfSONAR | ElasticSearch |
| latency      | 95th pct. last hour | 10 min. perfSONAR | ElasticSearch |
| mbps: fax: 1h | 95th pct. last hour | 5 min. FAX | ElasticSearch |
| mbps: fax: 1d | 95th pct. last day | 5 min. FAX | ElasticSearch |
| mbps: fts: 1h | 95th pct. last hour | 5 min. Rucio | ElasticSearch |
| mbps: fts: 1d | 95th pct. last day | 5 min. Rucio | ElasticSearch |
| mbps: dashb: 1h | mean last hour | 5 min. Rucio | Dashboard |
| mbps: dashb: 1d | mean last day | 5 min. Rucio | Dashboard |
| mbps: perfsonar | 95th pct. last hour | 10 min. perfSONAR | ElasticSearch |
| files: queued | count           | 5 min.       | Rucio    | Oracle         |
| files: done: 1h | count           | 5 min.       | Rucio    | ElasticSearch  |
| files: done: 6h | count           | 5 min.       | Rucio    | ElasticSearch  |
| files: queued-total | sum | -           | -        | convenience metric |
| files: done-total-1h | sum | -           | -        | convenience metric |
| files: done-total-6h | sum | -           | -        | convenience metric |

2. ATLAS Analytics

The ATLAS Analytics project focuses on creating systems which provide ATLAS Distributed Computing (ADC) with new capabilities for understanding distributed systems and overall operational performance [3]. These capabilities include, but are not limited to, the correlation of data from multiple systems (PanDA, Rucio, FTS, Dashboards, Tier0, PilotFactory, and more), predictive analytics to allow arbitrary data mining or machine learning algorithms over raw and aggregated data, and the ability to host new third party analytics services on a scalable compute platform. The project also aims to satisfy a variety of use cases for users doing ad-hoc analytics, as well as provide an open platform with documented collections and tools to broaden user participation and contribution.

The three main backends are Hadoop, ElasticSearch, and Jupyter, along with their respective analytics software. This includes frameworks like Spark, Python with numpy/scipy, Tensorflow, R, ROOT, and many more. The maintenance of these backends is currently shared between CERN IT and the University of Chicago. Across these backends the ATLAS Analytics project maintains ad-hoc data analysis notebooks upon requests from users, as well as dedicated analytics projects such as network metrics aggregation, perfSONAR[4] analytics services, the group space overview, reports for scrutiny groups, statistics for distributed analysis, and the data knowledge catalogue.

2.1. DDM network metrics

The DDM network metrics are prepared as a JSON document with source and destination sites as the unique key pair, as shown in Table 1. An updated metrics JSON is published every 5
minutes. The update intervals of the respective keys do not necessarily correspond with the backend update rate. For example, for a given link perfSONAR might only update once per hour but we still query the backends every few minutes because we cannot know when individual backend updates happen. More details about the perfSONAR measurements as used for this study is available elsewhere [5]. Most of the keys are self-explanatory, however, there are a few caveats. Closeness is an integer metric as defined by ADC Operations within AGIS[6], which ranks a given link based on human experience; a closeness of -1 means the link is explicitly blacklisted. The megabytes/second for FAX[7] show a distinct plateau characteristic, which has been identified as dynamic router reassignments. This leads to periods where potentially excellent links are downgraded automatically without ADC involvement. As this happens regularly, instead of automatically adjusting the values we decided to keep the values as-is to spot problems quickly. Additionally, all megabytes/second metrics are per-file and include the protocol handshake times. Especially for small files these handshakes can take longer than the actual time on the network. The exception is megabytes/second from perfSONAR, which represents the maximum achievable link throughput calculated by packetsize and RTT. The number of queued files reflects those waiting or active on the link per activity but only includes files where the sources have been resolved already. A relevant scenario might be a large queue of file transfers to a single badly connected data centre with a potential duration of multiple weeks. Instead of swarming the file transfer service, the sources for these transfers are only resolved immediately before submission. Finally, to reduce complexity of the JSON file the enqueing rates of the data management system Rucio[8], though available, are omitted.

2.2. DDM transactional state events

While the DDM network metrics describe the state of the network we also have realtime events that describe the transactional state of the ATLAS data management system Rucio. Every day approximately 10 million state changes are registered into ElasticSearch as a JSON dictionary across all Rucio activities. We collectively call them rucio-events. This includes events like transfer-done, deletion-queued, dataset-obsoleted or similar. For the purposes of this study, the transfer-done events are the most important. The content of a transfer-done event includes, but is not limited to, the following keys: event-type, request-id, transfer-id, created-at, submitted-at, started-at, finished-at, activity, filesize, checksum, source, source-type, destination, destination-type, transfer-duration, filename, transfer-protocol, exitcode, or transfertool.

These state events are widely distributed to update other ADC systems with changes from data management, for example, to notify the workflow management system that a dataset has finished transferring, or that new files have been created at a data centre. It presents a comprehensive view on the data management part of the experiment and can thus be used together with the network metrics to create a transfer time estimator.

3. Regression model

Decision tree learning is an effective and efficient tool for classification and regression of large datasets, and may find nonlinear relationships between variables [9]. Since decision trees are prone to overfitting, multiple random samples may be generated from the training data and fitted with separate decision trees. This method generates a forest of predictions that is averaged, producing a final prediction which is robust to outliers and noise. We use the Python library scikit-learn’s implementation of a random forest regressor, which can produce continuous predictions as an output, suitable for time series data.

Within rucio-events, each file has a size in bytes and an activity which depends on the file’s function, for example, user output or simulation input. Timestamps are recorded when the file is submitted to the queue, when the file begins transferring, and when the file has completed
transferring. The time-to-complete (TTC) data that is used as the dependent variable within the regression model is computed from these timestamps.

Instead of taking all files from the last four weeks, a random sample of 100 files per 10 minutes is gathered from rucio-events. This is done to avoid overfitting on intervals that contain many files submitted in a short period. Applying this sampling reduces the total size of the data considered by about one half.

Since certain activities are prioritized by the data transfer services, it is necessary to distinguish among activities when constructing the model. In order to prepare the activity data for regression, each value is coded as an n-vector, where n is the number of distinct activities. For each observation, the index corresponding to the appropriate category holds a value of 1, and the rest of the indices are zero. For example, in the case of a variable with 3 possible categories the first is encoded [1,0,0], the second [0,1,0], and the third [0,0,1].

From the DDM network metrics throughput measures are acquired as well as fields containing the number of files queued per activity. These fields are normalised to the smallest common bin width of 10 minutes, aggregated by sum for the queued fields and by mean for the throughput measures. To connect the rucio-events data to the DDM network metrics, each file from rucio-events is associated with the appropriate aggregated data bin from metrics by the file’s submitted at timestamp. The joined dataset matrix rows are then sorted according to increasing submitted at timestamp. Finally, any rows with missing values are not used for regression.

A training set is formed from the first 80pct of the cleaned and sorted data, while the last 20pct is reserved for validation. This choice of split is arbitrary: a 70/30 or 60/40 split could have been chosen, although preliminary results suggest that these choices cause an increase of outliers in predicted values. Fifty trees were chosen as the forest size. The final model is able to predict the TTC of a file, based on file size and file activity, including network status and performance at the time of file submission.

4. Regression tree evaluation

Figure 1 shows a direct example of the predictor. The red test data is the actual TTC of the transfer, and blue represents the model prediction for that particular transfer. At first glance, the model is capable to recognise short and long bursts, however a truly accurate prediction is not possible. High spikes, such as mid August 9th, are hinted at, but never captured at full intensity, others such as mid August 12th are completely missed. To evaluate these results in detail, we now focus on a single link.

Figure 2 shows the result per activity on the one-way link from the CERN data centre to Brookhaven National Laboratory (BNL)’s data centre. A few activities are unused, or have very low usage. Within each activity the model seems to capture bursts of small files quite well, as seen in activities Data Consolidation or Production Output. However, long transfers, with TTC reaching multiple tens of minutes, are never modelled correctly. The longer stream of large file transfers around August 10th in the T0 Export however is captured to some extent and picks up after a few minutes. The model thus seems to rely on a minimum amount of continuous characteristics in the transfer submissions.

The regression model performs adequately for all links, when looking at the details, however the results are nowhere near practical use. Additionally, because the regression model performs poorly on activities with low file counts, those activities often yield a considerably higher RMSE, for example, staging files from tape to disk. There are significant deviations at times. For example, the central plot in Figure 2, corresponding to the Production Input activity, displays high TTC for some data that is not captured by the model. This is especially evident in a histogram of the error difference, visible as a peak centred at approximately -140 minutes in the tail of the Production Input activity.

Figure 3 shows the error summary for the three main activities. The margin of the
RMSE covers almost 2 hours for T0 Export, several multi-modal errors of 50 minutes each for Production Input, and 1.5 hours for the Data Consolidation. Especially the multi-modality is quite surprising, given that Production Input should be a quite stable transfer activity. We have not yet been able to investigate the cause of this effect.

Because of the current exploratory nature of our work, it is impossible to determine whether all significant variables have been considered in the model. Indeed, since the spikes in the TTC data are not always captured by the model, and the model occasionally predicts spikes that do not occur, there is evidence that an important component is missing. There are two potential improvements to this model that come to mind. First, it is known that the transfer of data travelling in one direction along a link can affect the queue time of data flowing in the opposite direction, because of the read-write capabilities of the storage systems on either side. There is currently no measure for this influence, so it must be accepted as bias, but if in the future such a measure is implemented, it may be advantageous to add it to the model. Second, the current implementation of the regression model assumes that there is only one file being submitted at a given time. Knowledge of the number and size of files submitted simultaneously, or within a small time window, may improve the accuracy of this model, especially for large batch submissions.

5. First steps with deep learning
As an alternative approach to improve the predictions we investigated different types of artificial neural networks. Especially Long Short-Term Memory (LSTM) networks seem particularly suited to annotated time series data with recurring but non-periodic bursts [10]. We selected Keras as the software framework for the implementation since it supports LSTMs and opens the possibility for deep learning at later steps.

We created the LSTM with one hidden layer at 512 neurons and one dense layer with one output, activated by a rectified linear unit (ReLU). The training input variates are the same
as with the decision tree model, that is, source, destination, activity, bytes, start timestamp, and end timestamp of one transfer, using end timestamp as the label. We artificially limit the input to the LSTM to sequential training, to mimic the used learning process of the decision tree. The result of the model is shown in Figure 4. The model accuracy improved by one order of magnitude. The absolute prediction error has been reduced to less than 5 minutes across all activities and links. The multimodality previously observed still exists though, however it is now constrained within an acceptable error range.

To evaluate the deep learning capabilities of Keras we tried the same approach with a similar sized deep net, that is, 50 layers with 10 neurons each, again using ReLU activation. This is work in progress and does not represent its final state. The result are depicted in Figure 5 and show a worse performing model. The absolute errors are now comparable with the regression

**Figure 2.** Estimation of *Time To Complete* split by activity for the CERN to BNL link.
we were using the same selection of input variates as with the other models, and feeding them sequentially to the model. We are aware that this is suboptimal for a deep net because it should not depend on a user selecting variates, that is, feature engineering is not needed. Additionally, the LSTM gives us the possibility to learn historical data in bulk instead of sequential feeding. This bulk capability is as of yet also unused, due to the size of the history reaching multiple terabytes of hundreds of millions of events. There are ongoing plans to use a distributed TensorFlow installation to run large Keras LSTM models directly on Spark in early 2017.

6. Conclusion and future work
Placement of experiment data across computing sites is currently only based on fixed shares or random selection. To improve the data placement we need a way to quantify potential data distribution strategies so we can compare them. As a first step, we focused on the estimated

Figure 3. Absolute error across selected activities for the CERN to BNL link.

Figure 4. Single layer LSTM model absolute error across all activities and links.
duration of a file transfer for this quantification. Using the ATLAS Analytics platform as a central database of infrastructure and system metrics, we built a regression tree model which is able to train and estimate file transfer duration. The results were adequate, but delivered multimodal errors outside practical usefulness. For many activities and links the errors were in the order of multiple hours. Even under the assumption of queuing times of multiple days this straightforward approach seems unsatisfactory.

Our alternative approach using LSTM networks yielded an order of magnitude better results. The error is now well within a practical limit of 5 minutes, but still exhibits too much dispersion. As future work, we will now focus on improving the deep net using LSTM to better capture the non-periodic characteristics in our transfers.

References
[1] ATLAS Collaboration 2008, The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3 S08003
[2] Campana S 2015 J. Phys.: Conf. Ser. 664 032004
[3] ATLAS Analytics TWiki 2016 https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/ATLASAnalytics
[4] Campana S et al 2014 J. Phys.: Conf. Ser. 513 062008
[5] McKee S et al 2015 J. Phys.: Conf. Ser. 664 052003
[6] Anisenkov A et al 2014 J. Phys.: Conf. Ser. 513 032001
[7] Gardner R et al 2014 J. Phys.: Conf. Ser. 513 042049
[8] Garonne S et al 2014 J. Phys.: Conf. Ser. 513 042021
[9] Breiman L 2001 Machine Learning
[10] Hochreiter S and Schmidhuber J 1997 Neural Computation 9(8):1735-1780