EPJ Web of Conferences **214**, 06007 (2019) https://doi.org/10.1051/epjconf/201921406007
CHEP 2018

Application of Deep Learning on Integrating Prediction, Provenance, and Optimization

Malachi Schram[1,*], *Nathan* Tallent[1], *Ryan* Friese[1], *Alok* Singh[2], *Ilkay* Altintas[2]

[1]Pacific Northwest National Laboratory - Richland, WA, USA.
[2]University of California - San Diego, CA, USA.

Abstract. In this research, we investigated two approaches to detect job anomalies and/or contention for large scale computing efforts:

1. Preemptive job scheduling using binomial classification long short-term memory networks
2. Forecasting intra-node computing loads from the active jobs and additional job(s)

For approach 1, we achieved a 14% improvement in computational resources utilization and an overall classification accuracy of 85% on real tasks executed in a High Energy Physics computing workflow. For this paper, we present the preliminary results used in second approach.

1 Introduction

The current computing grid scheduling tools do not provide a robust mechanism to protect computing resources from "bad" behaviors. There are limited mechanisms to handle input/output (I/O), memory, and networking contention. In recent grid production campaigns, nearly 60% of the CPU usage was inefficiently used with up to 30% being blocked by I/O as shown in Figure 1. The motivation for this work is to build and study the use of a multi-tiered neural network model to identify and predict various sources of contention and proactively schedule jobs accordingly. The multi-tiered model would be built using the logical hardware hierarchy (cluster, node, job) as shown in Figure 2. In this paper, we investigate the ability to model the node level (single tier) computing load using a recurrent neural network.

2 High Energy Physics Distributed Computing

The current largest High Energy Physics (HEP) particle collider is the Large Hadron Collider (LHC)[1]. There are four large experiments that collect data and process data from these collisions. Each experiment has independently worked to solve the challenging task of computing workflows dealing with massive amounts of data. At the core of all four experiments, the workload management system (WMS) uses a "pull" model to assign jobs to resources. Below we briefly discuss computing models for all four experiments - ALICE, ATLAS, CMS, and LHCb

---

[*] e-mail: malachi.schram@pnnl.gov

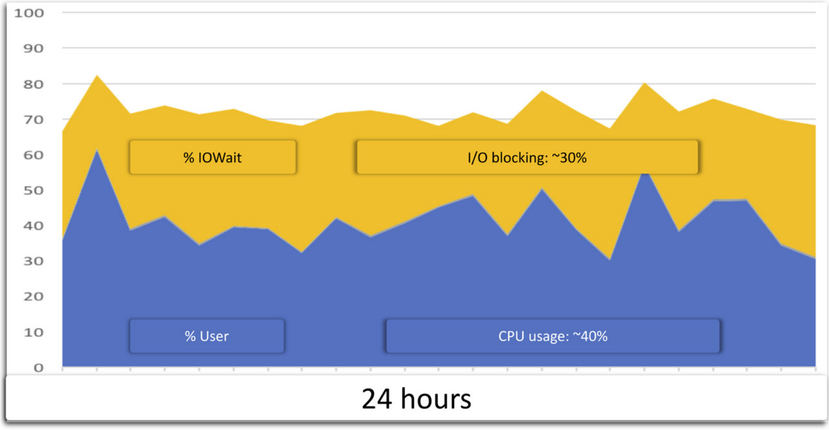

Figure 1: Example CPU utilization for a HEP computing workflow with problematic I/O usage.

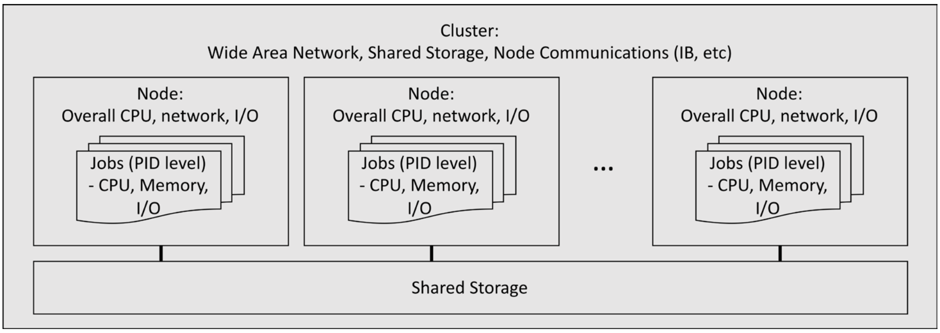
Cluster:
Wide Area Network, Shared Storage, Node Communications (IB, etc)
Node:
Overall CPU, network, I/O
Jobs (PID level)
- CPU, Memory,
I/O
...
Shared Storage

Figure 2: Multi-tiered cluster level schematic diagram of data sources.

The grid computing for the ALICE experiment is provided by AliEn. Its WMS uses Job Agents (JA) that optimizes using the following strategies: splitting jobs, so that each part can be submitted to different Computing Element (CE); replicating data, so that the jobs can be easily executed; requesting the installation of software packages on CE.

The distributed computing solution for ATLAS is called PanDA. The PanDA server receives work from users and production servers and places it in a global job queue, upon which the brokerage module operates to assign and prioritize work on the basis of job type, priority, input data and its locality, available CPU resources and other criteria. The allocation of job sets to sites is followed by the dispatch of the corresponding input datasets handled by a data service interacting with the ATLAS Distributed Data Management system.

The CMS experiment uses glideinWMS to provide a pilot factory for a Condor based WMS. CMS has a new dynamic Data Management System (DMS) that automatically places datasets at computing sites when they are created and then deletes them when they are no longer needed, based on "popularity" information. This allows for more agile and efficient use of disk space.

The LHCb computing model uses the DIRAC workflow. Pilot agents run on Grid Worker Nodes (WNs) reserving resource for the immediate use and requesting compute jobs from the WMS. Once the Pilot Agent arrives at a worker node the local environment is checked prior to any job scheduling. If any problems are discovered, such as a lack of local disk space or inconsistencies software environment, the Pilot can terminate gracefully and free the resource.

We find that at the core of the all above HEP distributed computing models a pull scheduling mechanism with late binding of resources to payloads is implemented with limited resource information. We are unaware of any production system that performs any anomaly detection and contention forecasting to improve efficiency and throughput.

3 Belle II computing use case

The Belle II collaboration was officially founded in December 2008 and has grown to include over 800 members at 108 institutions in 25 countries. With collaborators geographically located across North America, Asia, Europe, and Australia, distributed computing is required in order to fulfill the processing and storage demands of the collaboration.

Belle II has chosen DIRAC to provide core functionality for their distributed computing model [3]. DIRAC builds a layer between the users and the resources with systems that orchestrate the computing tasks that require specific types of resources. Heterogeneous computing resource providers can be seamlessly integrated into DIRAC using common interfaces. The Belle II computing infrastructure has leveraged and expanded this feature to access various resources.The Belle II collaboration has developed a dedicated extension on top of the core DIRAC framework. This extension provides dedicated tools for users, modified job wrapper code to work with the Belle II software framework, and numerous systems with components that satisfy specific computing and data workflows required by the Belle II collaboration.

The Belle II production group coordinates the upcoming Monte Carlo and data processing campaigns and submits the jobs to the BelleDIRAC system. These computing jobs are scheduled based on the available resources and proximity to any required input datasets. Similar to the LHC computing model, there is not scheduling mechanism to account for any local (node or cluster level) or large scale contention should it arise. Unfortunately, these large scale contention problems do arise and cause non-linear reduction in the overall throughput of the computing distributed system.

4 Demonstrator and Dataset

For our studies, we extended the DIRAC system to capture additional metrics, such as I/O and Wide Area Network, at a 1Hz sampling rate. We deployed three dedicated compute clusters with different hardware architectures and configurations to study the variability of identical compute workload on different setups. We plan to use these compute clusters to study macroscopic contention, such cluster level wide area network access and shared storage. For this paper, we focused on the node level information for one cluster. The test cluster for the demonstrator was composed of 64 Intel nodes with 48 threads providing 3072 compute slots. We setup a dedicated DIRAC SiteDirector[1] agent to provide direct access to the HTCondor queue. All DIRAC agents related to the WMS were set to the shortest pulling cycle (1 minute) to ensure that jobs were quickly pushed to the queue.

We ran the demonstrator for 48 hours with continuous Belle II Monte Carlo jobs. The compute hardware configuration and job mapping metrics were captured using the DIRAC system in the JobDB database [2]. The additional performance metrics were captured in the dedicated extension and the JobID was used to relate the information between the DIRAC system and the extension. Figure 3 illustrates some of the traces captured for one node during the Belle II Monte Carlo demonstrator study.

5 Neural-Net Training and Performance

We developed a long short-term memory (LSTM) recurrent network [4] to predict the compute load using Keras 2.1.2 [5] and TensorFlow 1.5.0 [6]. We used the time series information from CPU utilization (usr, iowait, etc.), jobs per node, and memory as our input feature vector. We prepared a batch generator to create 256 batches with 20 minute traces and modified the loss function to provide a 5 minute "warm-up" training phase. The training results are shown in Figure 4 with the forecasting time set to 1 minute.

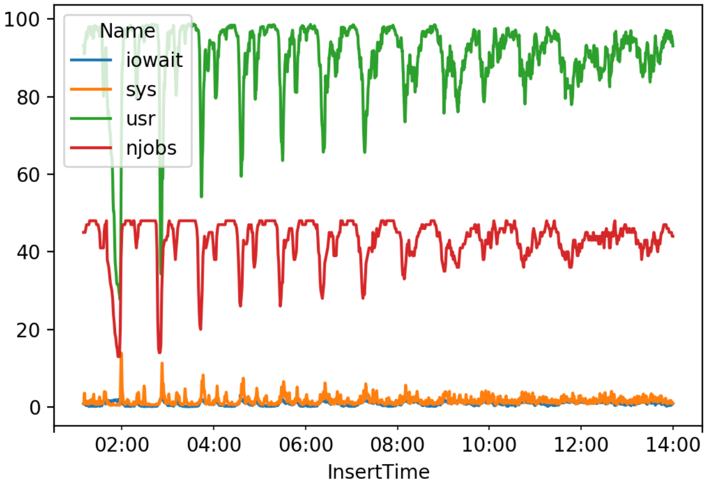

Figure 3: Example CPU utilization for a HEP computing workflow during the Belle II Monte Carlo demonstrator study.

6 Conclusions

In this paper, we present some preliminary work on the use of recurrent neural network to predict the computing load on a single node in a distributed computing environment. We

---

[1] A SiteDirector is an agent that retrieves jobs from the master DIRAC WMS.

[2] We plan to use this information in the multi-tiered neural network model of the distributed system.

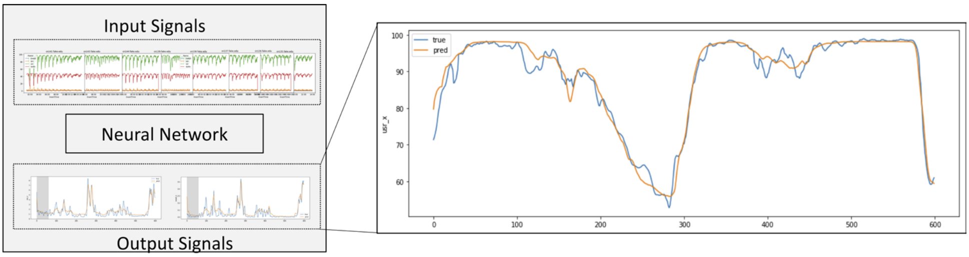

Figure 4: Example prediction of the CPU utilization traces for a HEP computing workflow.

created a multi-cluster demonstrator to mimic the Belle II distributed computing model and added some extension to the DIRAC system to capture detailed computing metrics. We ran a mock Belle II Monte Carlo campaign on 3074 dedicated compute slots. We create and trained a LSTM model to predict the compute load on each node based on the metrics capture during the campaign. We find for a single production job type we can predict the CPU load to between than 10%. In the future, we plan to conduct a hyper-parameter scan, enhance the network model, and train our model on additional production job types. Specifically, we will focus on known contrasting jobs type (CPU vs. memory vs. I/O). Additionally, we will deploy and evaluate the use of our LSTM model in an online setting to perform real time forecasting and anomaly detection.

References

[1] L. Evans and P. Bryant, *"LHC Machine"*, JINST **3**, S08001 (2008). doi:10.1088/1748-0221/3/08/S08001

[2] A. Tsaregorodtsev *et al.*, *"DIRAC: A community grid solution"*, J. Phys. Conf. Ser. **119**, 062048 (2008). doi:10.1088/1742-6596/119/6/062048

[3] Takanori HARA and Belle II computing group, *"Computing at the Belle II experiment"*, J. Phys. Conf. Ser. bf 664, 012002, 2015

[4] Sepp Hochreiter; Jürgen Schmidhuber *"Long short-term memory"*, Neural Computation. 9 (8): 1735-1780. (1997) doi:10.1162/neco.1997.9.8.1735.

[5] Chollet, François *et al.*, *"Keras"*, GitHub, https://github.com/fchollet/keras, 2015

[6] Martín Abadi *et al.*, *"TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems"*, http://tensorflow.org/, 2015