Seasonal study of user demand and IT system usage in datacenters [*]

Damien Landré[1,2], Laurent Philippe[1], Jean-Marc Pierson[2]

[1] *Université de Franche-Comté, Institut FEMTO-ST, Besançon, France*
[2] *Université Toulouse 3, IRIT, Toulouse, France*
damien.landre@femto-st.fr, laurent.philippe@univ-fcomte.fr, Jean-Marc.Pierson@irit.fr

Abstract: To limit the impact of datacenters on climate change various sustainable and effective solutions are being developed. The DATAZERO2 project aims to design datacenters running solely on local renewable energy combined with storage sources. A multi-hour prevision of user demand (workload) or of IT system usage, using time series, are possible ways to plan energy usage, with the aim of reducing energy consumption while maintaining a required quality of service. The seasonal component of time series is a key element, since it indicates one or more periodic phenomena that can be estimated and implemented in forecasting methods. In this paper, we propose a seasonal study of multiple time series from different workloads. Seasonality is studied using periodograms, a highly effective Fast Fourier Transform technique for detecting seasonal patterns. In addition, since the detected seasonal patterns can be irregular (unexpected, delayed and/or extended user demands or IT system usage), we propose a distribution-based clustering using the Kruskal-Wallis test and a seasonal peak appearances analysis. The results show that most of these time series are multi-seasonal (daily and weekly seasonal patterns) but highly irregular, which can reduce the performance of seasonal forecasting methods.

Keywords: Workload characterization, Workload prevision, Datacenter

1 Introduction

Over the past decade, the datacenter workload have increased by a factor 10. This makes them one of the most energy-intensive parts of the ICT [20], accounting for 1 to 1.5% of global energy consumption. In order to limit their potential explosion in energy consumption, enormous efforts have been made, such as improving the efficiency of electronic components, cooling, power leakage, *etc.*. Another way of reducing energy consumption is to predict user demand to only switch-on machines on demand. Workload prediction is key issue in proactive resource management and auto-scaling. Accurate prediction of workload is of high importance to mitigate energy consumption, while maintaining the required quality of service (QoS) level [25]. This is even more important when the datacenter is completely off-grid, in order to efficiently manage renewable energy sources and storage devices.

The DATAZERO [29] (2015-2019) and DATAZERO2 projects (2020-2024) aim to design and operate a datacenter only fueled by renewable energy. The orchestration between the power and energy management (power production and energy storage) and the IT management (workload processing and servers states) is based on a negotiation process [46]. This process uses workload and power production previsions to determine a power command, that will be applied to the infrastructure during the next time window, typically the coming 72 hours. A power command is a time series of power values for each time interval of the time window. In order to operate, the negotiation process needs predictions of the IT power needs during the given time window. Furthermore, in order to take full advantages of several energy storage devices (like batteries and hydrogen tank), the prediction must reflect the seasonality of the demand: indeed, when workloads peaks and troughs are known to occur at given instants in the future, smarter decisions can be taken to anticipate them, for instance by using batteries on the short term and hydrogen on the long term.

[*]This work was supported by the DATAZERO2 project ("ANR-19-CE25-0016") and the EIPHI Graduate school ("ANR-17-EURE-0002").

In this work, we propose and assess a methodology to identify seasanality in workloads. To highlight the seasonal pattern(s) present in each time series, we use a periodogram, a FFT (Fast Fourier Transform) technique that estimates the spectral power of each frequency. We then analyze the irregularities of each detected seasonal pattern. These irregularities are evaluated using: (i) a distribution-based clustering using the Kruskal-Wallis test, (ii) an analysis of the frequency of occurrence of the seasonal peaks.

This paper contributes with a suggestion for using periodograms to detect seasonal pattern(s) in a time series, instead of box-plots, automatic decomposition and ACF (Auto-Correlation Function). To estimate the irregularity of a seasonal pattern detected in a time series, we propose an algorithm based on the Kruskal-Wallis test for clustering the periods We also propose two forecasting models for datacenters: a multi-hour prevision of system usage based on allocated cores per hour time series, and a multi-hour prevision of user demand based on several time series.

In the following, related work is detailed in Section 2. The problem definition and objectives are formally defined in Section 3. Section 4 describes the methodology used to identify and assess seasonality. Section 5 presents the data sets used and the filters applied. Section 6 presents the seasonal study of time series for different workloads, *i.e.* detected seasonal patterns using periodograms and irregularities using a distribution-based clustering and frequency of occurrence of seasonal peaks depending on time. Finally, Section 7 gives a characterization use case and Section 8 summarizes the paper, highlighting the main conclusions and future work.

2 Related Work

In this section we consider the work related to the workload characterization, then more specifically to seasonal studies. Finally, we review the application of workload characterization and seasonality through forecasting models and/or methods.

2.1 Workload characterization

The literature abounds in statistical studies using descriptive methods to analyze cloud and HPC workloads to improve scheduling, virtual machine allocation, to generate synthetic workloads close to those observed or to select the best forecasting methods. In [6], the authors present the state of the art of workload characterization in various application domains. The cited articles that consider cloud workloads mainly focus on jobs/tasks and virtual machines, and the most relevant research questions investigated refer to resource usage, job arrival and workload pattern. In addition, they present the methodological approaches and different modeling techniques employed to characterize workloads. Di et al. [11] study the differences between a cloud datacenter and Grid/HPC systems. Several statistics about jobs and machines are analyzed and compared. Google jobs are submitted with much higher, more stable frequency and lower resource demand for CPU and memory than that of Grid/HPC jobs. Moreover, noise of CPU load is 20 times higher and CPU usage is lower and less stable than in Grid/HPC systems. In [30], as HPC workloads are becoming more diverse at both application and job levels, the authors present a methodology to characterize workloads and assess their heterogeneity to improve schedulers in these systems. They assess several jobs characteristics by using methodologies from Feitelson's book [14]. Jobs, queue and performance are characterized and compared with other HPC systems. In particular, they observed that 90% of the jobs have inter-arrival times under two minutes and 86-88% run for fewer than two hours. Also, a trend analysis of the various workload parameters is given. In [49], the authors propose a method to model the workload in HPC clusters and generate synthetic workloads. In [21] the authors analyze important variables relevant to resource utilization, the number of CPU cores and VMs used by considering Physical Machine (PM) level metrics in the context of cloud datacenters.

Clustering is one of the methods used to characterize, describe and analyse workloads according to different metrics with the same objectives as mentioned above. Sindhu et al. [34] propose a generative Dirichlet process-based model to characterize cloud workloads This process is performed in several distinct steps to generate a synthetic workload. First, they identify the number of task domains that generate the workload, according to different metrics. Then the synthetic workload, based on the task domains, is compared with the real workload. Other classification methods are also applied in [26], [10], [4], [28] and [38] with Google, Alibaba, DKRZ and K computer workload datasets, respectively. Similarly, in [33], the authors use and compare 7 different machine learning algorithms to classify and characterize cloud datasets based on resource usage, to improve capacity planning, resource allocation and task scheduling. The more significant key factor are task duration and CPU usage. In [3], in the case of HPC systems, the authors experiment and employ a number of different feature selection methods and clustering algorithms. The results show that the test dataset could be best divided into three clusters.

Overall, these studies characterize Cloud and HPC workloads using descriptive statistics, trend analysis and clustering, but do not tackle the characterization of seasonality patterns.

2.2 Seasonal study

To detect seasonality, several methods are used in the literature. Trend and seasonality of the Wikimedia's workload are studied over 5 years in [13] by using descriptive statistics, time series analysis and polynomial splines. They use a simple time series decomposition and boxplots to highlight trend and seasonality. The request flow time series from the Wikimedia's workload clearly shows strong daily and weekly repeating patterns. In [5], the authors characterize the workload of an engineering corporation (aircraft manufacturer) over a period of 30 months. Distinctive and strong daily and weekly submission seasonal patterns and grid utilisation are observed by using box-plots, which is not enough to confirm seasonality. Task volume (execution time and number of cores) and dependencies are also considered to propose an algorithm that creates jobs based on observed patterns. In [35], the authors propose a seasonal and a non-seasonal study for time series analysis. In the case of the seasonal study, they propose to use ACF and PACF (Partial Autocorrelation Function) plots to obtain information regarding seasonality, trends and stationarity. But, as mentioned in [27], the FFT technique (such as periodogram) outperformed the ACF because the latter has restrictions in identifying the exact start time of the seasonality, and the seasonality must be significant before the ACF is able to detect it. The FFT technique is briefly used in [30] to detect the dominant submission seasonal pattern of the time series of the jobs submitted per hour, with a daily and weekly repeating patterns corresponding to user working periods, and three and six months patterns which are related to the allocation rules set by NERSC. However, they do not mention that the stationarity of the time series has been tested.

Although addressing the detection of strong seasonality, most of these studies only use simple methods such as (i) box-plots, which lose the dependency between values, (ii) automatic decomposition, which is only efficient with strong seasonality, or (iii) ACFs, which is complicated to analyse when the time series presents several seasonality or an insufficiently strong seasonality. Moreover, it is done only for one time series derived from one or two workloads. Being designed to detect periodicity in behaviors, FFT are suitable to detect one or more strong or weak seasonal patterns.

2.3 Prevision

There are many studies on workload previsions in the cloud and HPC fields [25], [19] and some of them use seasonal patterns and propose adapted forecasting methods. This is the case in [35] where the authors propose an adaptive prediction model (called TASM) using linear regression, ARIMA, and support vector regression (SVR) for web applications in the context of cloud computing. One of these methods is selected depending on the workload properties (linear slow, fast and non-linear slow scale data, respectively) identified by a workload classifier, and parameters through a heuristic. They experiment their model with NASA and ClarkNet data by predicting step by step a single value. As a result, TASM is better than the others (ARIMA, SVR, LR, Naïve) with a MAPE (Mean Absolute Percentage Error) of 14% and 11%, respectively. Higginson et al. [18] propose an approach to reduce the complexity of performing a prevision and to produce accurate predictions for complex workloads by investigating the applicability of different time series modelling techniques with differing levels of seasonality and growth. First, this approach performs a linear interpolation if there are missing data, then it analyzes time series data (trends, seasonality...). Finally, each method (TBATS, HES, ARIMA, *etc.*) is computed to obtain an RMSE (Root Mean Squared Error). In [32], the authors propose a prediction algorithm based on Holt-Winters exponential smoothing method to model cloud workload with multi-seasonal patterns by adding seasonal indices and smoothing equation for each seasonal pattern, and by employing an Artificial Bee Colony algorithm to optimize its parameters. To assess it, they use many real workload traces, but only one is shown, namely the Saskatchewan Log that contains HTTP requests. The time series considered from this log is the number of required CPU. As a result, the algorithm outperforms other exponential smoothing methods in terms of MAPE, RMSE and percentage of predictions within 25%.

Finally, we found that, in the literature, the methods used to detect the seasonality in workloads are not sufficient since multi-seasonality is not studied and fully exploited. Moreover, they focus mainly on short-term workload previsions (over the next hour) for online modules. There are an increasing number of suitable forecasting methods: BATS and TBATS [9] (2011), HES with multi-seasonal pattern [32] (2017), Prophet [37] (2018) or MSTL [2] (2021) to mention just a few.

3 Problem definition and objectives

3.1 Problem definition

Workload forecasting is one of the most frequently used ways of improving datacenter efficiency while reducing energy consumption. In addition, when the datacenter is completely disconnected from the grid, as in the Datazero2 project, this helps to maintain its robustness over the long-term. For the IT management, a workload prevision is needed to anticipate the power consumption of the datacenter and to plan the switch-on and -off of the machines, while maintaining a QoS level, *i.e.* not rejecting too much workload.

As seen in the previous section, the literature focuses mainly on short-term workload previsions for online modules. In our case, the IT management of the datacenter negotiates offline with the electrical management and requires a long-term workload prevision to generate a power consumption prevision. This can be done using time series previsions. We propose two forecasting models:

- The proposed IT system usage forecasting model consists of a time series of the maximum number of cores allocated by the system per hour as input to the forecasting method. The maximum number of system allocated cores has been chosen to ensure that the IT management of the datacenter switches-on a sufficient number of machines to maintain a given QoS level. The forecasting method returns a prevision for the next 3 days by steps of 1 hour. This forecasting model does not require any specific knowledge of the user demand, but only how the machines in the datacenter have been used in the past.
- The proposed user demand forecasting model uses multiple time series as input: number of submitted jobs per hour, work volume per hour (defined later) and maximum number of requested cores per hour. These are predicted using a forecasting method adapted to each of them. The previsions are then used to generate multiple possible scenarios of user demand, based on the confidence intervals given on the previsions. Fig. **??** illustrates how each scenario is created. For each scenario, jobs are created based on the number of submitted jobs prevision and placed according to the inter-arrival time distribution (Step 1). Their characteristics are then generated using the runtime distribution used to give them a deadline (Step 2), work volume (Step 3) and a number of requested cores (Step 4). The amount of work volume is split between the different jobs. Finally, these steps are repeated to generate other workload scenarios (Step 5), taking into account the uncertainty of the previsions.

The work volume per hour represents the volume of instructions to be processed. It is computed, for each hour, by the sum of the product of the runtime and the number of requested cores of the jobs submitted during this hour (see formula (1)), where n is the time series size in hour, m the number of workload jobs over the time period considered, s_j the submit time of job j in second, nc_j the number of cores required by the job and r_j the runtime of the job in second. The function $\mathbb{1}_h(s_j)$ is 1 if $s_j \in h = [3600(t-1); 3600t[$, and 0 otherwise.

$$\forall t \in [1, ..., n], work_t = \sum_{j=1}^{m} \mathbb{1}_h(s_j) nc_j r_j \tag{1}$$

3.2 Objectives

To improve the prevision quality, it is essential to know the seasonal components of the time series, especially for long-term previsions. Note that this study can also be used to improve scheduling in systems and/or to improve the production of synthetic workloads. Some seasonal patterns are however

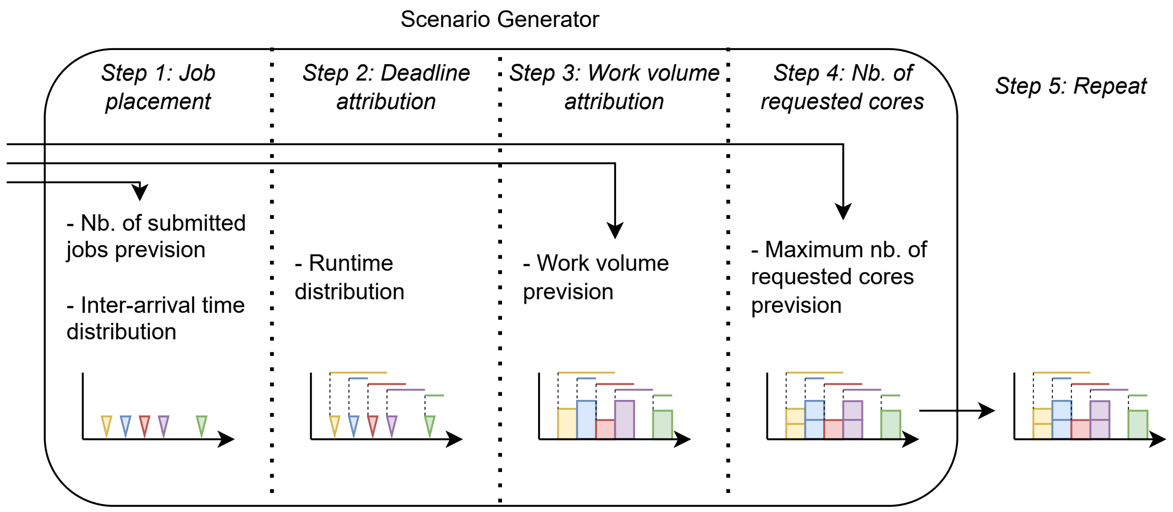

Scenario Generator
Step 1: Job placement
Step 2: Deadline attribution
Step 3: Work volume attribution
Step 4: Nb. of requested cores
Step 5: Repeat
- Nb. of submitted jobs prevision
- Inter-arrival time distribution
- Runtime distribution
- Work volume prevision
- Maximum nb. of requested cores prevision

Figure 1: Workload scenario generator steps: Job placement (step 1), deadline attribution for each job (step 2), work volume attribution for each job (step 3), number of requested cores for each job (step 4), generate a new workload scenario (step 5).

less obvious than others. They can be more or less irregular: peaks and troughs may not always appear at the same moment, the amplitude of peaks may be stronger or weaker, peaks may be longer or shorter, *etc.*. In addition, knowing whether or not one or more seasonal patterns are present in time series helps to identify which forecasting methods are more suitable than others. For example, ARMA and exponential smoothing technique are only suited for modeling single seasonality, and unable to account for multiple seasonal patterns [48], except for Holt-Winter, which has been adapted in [32] to consider multi-seasonality. In contrast, Prophet and TBATS are suitable for time series with multiple seasonality. But seasonal irregularities can potentially influence these forecasting methods and compromise their performance. For these reasons, the objectives of this paper is to study the seasonality of time series used in the forecasting models and their potential seasonal irregularities.

4 Methodology

The methodology we propose in order to analyse the seasonality present in a given workload dataset and the suitability of the time series for prevision is the following:

1. Filter the dataset to remove corner cases
2. Identify seasonality using periodograms
3. Assess the irregularity of time series seasonal patterns, using a distribution-based clustering using Kruskal-Wallis test
4. Check for seasonality peaks appearance, examining the frequency of occurence depending on time.

This section details steps 2 to 4, while the data filtering will be presented along with the dataset in the following section.

4.1 Identification of seasonality

To identify the seasonality of time series, we use periodograms, an FFT technique, as time series can be view as a sum of cosine waves with varying amplitudes and frequencies [42]. As a result, a time series can be expressed by (2).

$$\forall t \in [1, n], x_t = \sum_{j=1}^{n/2} \beta_1(\frac{j}{n}) \cos(2\pi\omega_j t) + \beta_2(\frac{j}{n}) \sin(2\pi\omega_j t) \quad (2)$$

Where n is the number of observed data item, $\omega_j = j/n$ the frequencies set and $\beta_1(\frac{j}{n})$ and $\beta_2(\frac{j}{n})$ the regression parameters. For each frequency, the regression parameters are estimated. Then the amplitude (spectral power) is given by (3).

$$\forall j \in [1, \frac{n}{2}], P(\frac{j}{n}) = \hat{\beta_1}^2(\frac{j}{n}) + \hat{\beta_2}^2(\frac{j}{n}) \quad (3)$$

A large value of $P(\frac{j}{n})$ indicates more importance for the frequency j/n in explaining the oscillation in the observed time series. Before applying a periodogram to a time series, it is necessary to check that it is stationary, *i.e.* mean, variance and covariance do not change over time. This is performed using statistical tests such as Augmented Dickey Fuller (ADF) test [12] and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test [24]. The main difference is that ADF test uses null hypothesis that a series contains a unit root (stationary or trend stationary if not), while KPSS test uses null hypothesis that the series is stationary (unit root if not). We use the ADF and KPSS tests by considering that a P-value lower than an empirical α threshold of 0.05 for ADF and upper than 0.05 for KPSS denotes that the time series is stationary. If the time series is not stationary, this may be caused by a trend (in this case, we use the Locally Weighted Scatterplot Smoothing, LOWESS [7] to remove the trend) or a strong variation in values (in this case, we differentiate the time series *i.e.* by substracting from each value of the series the value that precedes it to obtain a stationary series).

4.2 Assessing the irregularity of time series

To assess the irregularity of the time series, we check if seasonal periods derive from the same distribution. If so, this means that the values observed for each sample span over the same interval, and that peaks, troughs and intermediate values have the same order of magnitude. This is an important point in assessing the suitability of seasonal forecasting methods. if the time series includes a trend, it is removed beforehand using LOWESS to study seasonal periods around the trend and prevent it from affecting the distribution of seasonal periods.

The Kruskal-Wallis test [23] is used to check if two or more given samples are significantly different or not from a distribution point of view. The test is significant if the samples are derived from the same distribution. However, if the test is not significant, it does not identify the sample(s) derived from one or more different distributions. One can use a Conover-Iman post-hoc test [8], which is a test based on the same principle as the Kruskal-Wallis test, by comparing samples in pairs. But even if it indicates the sample(s) contributing to the insignificance of the Kruskal-Wallis test, it does not indicate how many different distributions are present in the time series.

For this reason, we propose a distribution-based clustering using the Kruskal-Wallis test (see Algorithm **??**). It is computed if the Kruskal-Wallis test applied to all samples of the seasonal pattern is not significant and if the Conover-Iman test indicates that more than one sample is concerned (using an empirical α threshold of 0.05 for both tests). This clustering process is simple: the algorithm takes as input the set of samples $\mathcal{D}$ and the α threshold is initiated by creating a set of clusters $\mathcal{C}$ and a cluster $\mathcal{C}_1$ containing the first sample (line 2 to 4). Then, starting with the second sample, for each sample, the Kruskal-Wallis test is applied between the sample and each cluster. If the test is significant for one cluster, this means that the sample and those in the cluster are derived from the same distribution. In this case, the sample is allocated to the cluster concerned (line 6 to 12). On the other hand, if the test is not significant with all clusters, this means that the sample is not derived from the same distribution as those of the clusters tested. If so, a new cluster is created and the sample is allocated to the latter (line 13 to 14). This process ensures that the samples in a given cluster are derived from the same distribution, and that each cluster is different from the others.

Finally, each cluster is composed of samples having the same distribution. The number of clusters thus indicates the level of irregularity of the time series. From one seasonal pattern to another, values may differ considerably with the number of clusters.

Algorithm 1: Distribution-based clustering algorithm

```
Data: $\mathcal{D}, \alpha$
Result: $\mathcal{C}$
begin
    $\mathcal{C} \leftarrow \emptyset$
    for $d_i \in \mathcal{D}$ do
        if $\mathcal{C} = \emptyset$ then $\mathcal{C}_1 \leftarrow \{d_i\}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_1$
        else
            $allocation \leftarrow False$; $m \leftarrow card\{C\}$
            for $j \leftarrow 1$ to $m$ do
                $p \leftarrow KWtest(\mathcal{C}_j,\ d_i)$
                if $p \geq \alpha$ then
                    $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup \{d_i\}$
                    $allocation \leftarrow True$
                    break
            if not $allocation$ then $\mathcal{C}_{m+1} \leftarrow \{d_i\}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{m+1}$
```

4.3 Checking for seasonality peaks

When seasonality is regular, both peaks and troughs appear at the same time, making forecasting easier. In the opposite case, it can be more complicated for forecasting methods to accurately predict the appearance of these peaks. We investigate the appearance of seasonality peaks to assess their recurrence, the results in section 6 showing that time series, for a given seasonal pattern, present several distributions considering the number of clusters. This is done by studying manually the frequency of occurence of seasonal peaks depending on the time of day and the day of the week, while the results show that time series are mainly daily and weekly seasonal.

4.4 Reproducibility

The source code and the complete set of experimental data and resulting figures is available in open access on the Git of the research team for reproducibility[1].

5 Data sets used

5.1 Presentation

We study the seasonality of four time series from six real workload logs from the Parallel Workloads Archive. The time series derived from these workloads are the number of system allocated cores per hour, the number of requested cores per hour, the work volume per hour and the number of submitted jobs per hour. The purpose of the first time series is to predict the system utilization. The purpose of the three other time series is to generate a workload prevision.

We study HPC workloads from the Parallel Workloads Archive. We made this choice because cloud workloads such as Google Traces [47] or Alibaba Traces [1] do not last enough time to confirm seasonal patterns, except for the 5-year Wikimedia's workload, which has already been studied [13] but is assessed in this work considering the request flow and the number of bytes per hour time series.

The first workload log, called Ciemat Euler [31], [40] consists of 9 263 012 jobs from the Euler supercomputer located at the Centre for Energy, Environment, and Technology Research in Madrid, Spain. The recording period is approximately 9 years, from November 2008 to December 2017. In November 2008 the cluster were composed of 144 nodes. In November 2009, 96 more nodes were added and, until the end of the recording period, the configuration is 240 blade nodes for a total of 1920 cores. All nodes are composed of dual Xeon 4-core 2.96 GHz or 3.0 GHz processors and 2 GB RAM per core. Job runtimes are mainly less than 10 hours, with CPU node utilization mostly between 20% and 90%. Note that the utilization does not exceed 60% for the first two years as new nodes were added in November 2009, before reaching 100% locally for 3 years and then decreasing to a maximum of 80%.

[1] https://gitlab.irit.fr/sepia-pub/open-science/seasonal-study-of-user-demand-and-it-system-usage-in-datacenters

The second log, called ANL (Argonne National Laboratory) Intrepid [39], [36], is a record, from January to September 2009, of 68 936 jobs from a large Blue Gene/P system called Intrepid. It consists of 40 960 4-core nodes with 2GB per node for a total of 163 840 cores and remains the same during the entire duration of the recording. As for the CIEMAT Euler workload log, job runtimes are mainly under 10 hours, but with node utilization mostly between 0 and 100%. Note that the utilization exceeds 100% several times. This is possibly due to the application of resource constraints by the scheduler [15].

The third log, called PIK IPLEX [43], contains more than 3 years (April 2009 to July 2012) worth of accounting records from the 320-node IBM iDataPlex cluster at the Potsdam Institute for Climate Impact Research (PIK) in Germany, for a total of 742 964 jobs. Each node is composed of 8 cores and 32 GB of memory. In the whole system, there are 2 560 cores. Job runtimes are mainly under 4 days and can reach up to 30 days. As for the ANL Intrepid workload log, the utilization can exceed 100% several times for the same reason.

The fourth workload log, called RICC [44], consists of 447 794 jobs from May to September 2010. It contains several months worth of accounting records from the RICC (RIKEN Integrated Cluster of Clusters) in Japan, from a cluster of 1024 nodes, each with 12 GB of memory and two 4-cores CPUs.

The fifth log, called MetaCentrum [41], [22], consists of 5 731 100 jobs from January 2013 to April 2015. Note that the scheduling system was significantly reconfigured in the middle of this period and some nodes change. MetaCentrum is composed of a varying number of heterogeneous clusters with between 4 and 384 cores and between 3 and 6144 GB of memory per machine, depending on the cluster. In most cases, job runtimes are less than 10 hours, and can reach up to 30 days. Regarding the utilization, we note an increase due to system configuration changes and scheduler reconfiguration.

The last log, called UniLu Gaia [45], consists of 51 987 jobs from May 2014 to August 2014 used for large data problems and physical simulations. These data originate from the Gaia cluster at the University of Luxemburg. Gaia is a heterogeneous cluster that has been upgraded several times, with a total of 151 nodes and 2004 cores. Over 50% of jobs have a runtime of 10 minutes or less. The utilization varies mainly between 20% and 80%.

All workload traces include information on jobs but do not include information about applications nor the input data.

5.2 Filters

Since the clusters were already running before starting logging their activity and after stopping logging, some jobs that were using the resources are not present in the original log, *i.e.* jobs started before logging and jobs finishing after the end of the log. For this reason, we remove the time window corresponding to the longest recorded job from both ends of the recording period. This filter is applied to all workloads and all time series to compare them with each other later in terms of irregularity. This allows us to keep only the time period during which all jobs currently being processed are known and minimize the time during which jobs are being processed but not recorded. For example, in the Metacentrum workload log, jobs submitted before January 1, 2013 run on the machines, but are not mentioned in the log. The same goes for jobs submitted before April 17, 2015 and not finished before this date. The longest job recorded in the log is 5045 hours (wait time + runtime). So we decide to keep only the jobs from July 30, 2013 to September 19, 2014.

Also, to be closer to a cloud datacenter and to maintain the same machine configuration over the entire period, a second filter is applied only to the Metacentrum workload to obtain another workload, called Filtered Metacentrum, by removing jobs run on specific architecture or hardware (*i.e.* large number of CPU, GPU, *etc.*) [16]. A subset of seven of the partitions (number 7, 9, 11, 14, 15, 18 and 19) of the machines is kept, with servers of 12-16 CPU cores and 12-134 GB memory.

After applying these filters, the ANL Intrepid workload log retains 83% of jobs and 4474 hours, the Ciemat Euler workload log retains 94% of jobs and 73903 hours, 71% and 24% of jobs and 9985 and 9538 hours for the Metacentrum and the Filtered Metacentrum workload logs, respectively. The PIK IPLEX workload log retains 93% of jobs and 26005 hours, and only 22% and 25% of jobs and 1198 and 1138 hours for the RICC and the UniLu Gaia workload logs. Note that these filters are not applied to the Wikimedia's workload log, for which we have no information on requests or their processing.

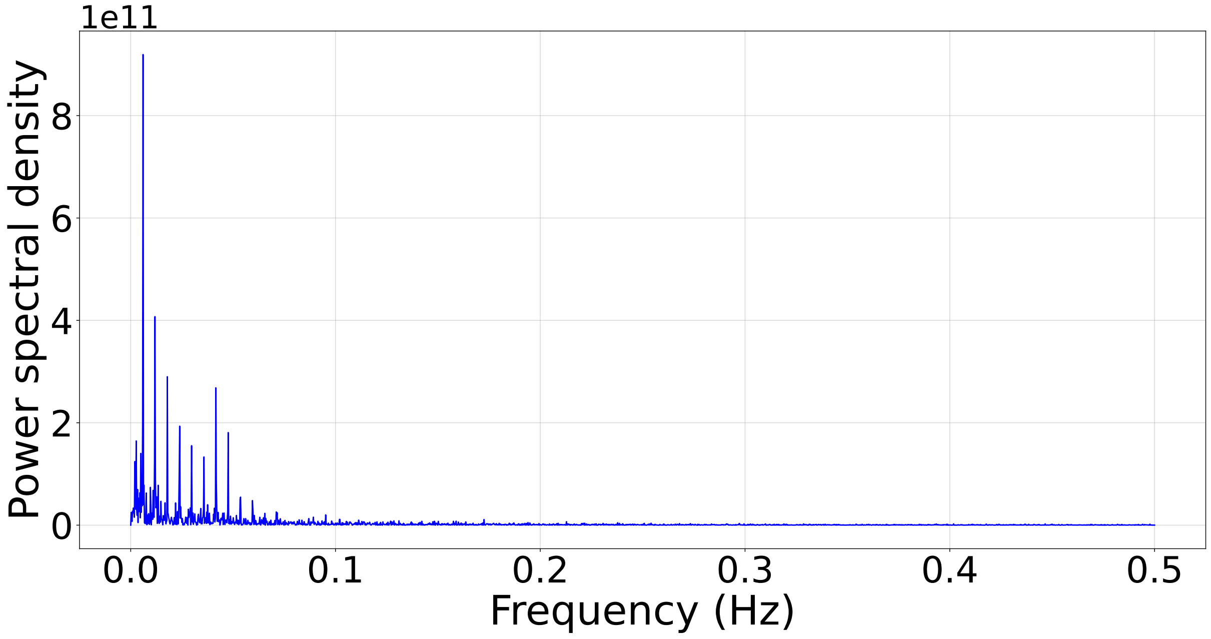


Figure 2: ANL Intrepid - Number of system allocated cores

6 Seasonal characterization

6.1 Seasonal periods

Firstly, we identify the seasonality present in the time series by applying the periodogram defined in Section 4. First, we check that time series are stationary according to the KPSS and ADF tests. If not, we remove the trend, or differentiate.

Fig. **??** shows for instance the periodogram of a ANL Intrepid workload time series. The figure presents the appearance of frequencies in the time series, the higher the spectral power the more present is this frequency. Note that peaks for which the number of seasonal patterns observed is less than 6 are not considered [17], which corresponds to the minimum sample size required to do a time series analysis. Peaks appear at different low frequencies depending on workloads and time series. These frequencies mostly correspond to daily ($\omega_j \simeq 0.04$Hz) and weekly ($\omega_j \simeq 6.0 \times 10^{-3}$Hz) seasonality, as can be seen on the figure.

Table 1 shows the power spectral density values (or amplitude), explained by (3), obtained from different workloads and for different periods. These values indicate how strongly frequencies are represented in the time series. Periods with significant peaks are indicated and the strongest peaks are highlighted. All time series are mainly based on a daily period and the majority of time series have two or more seasonal periods. However, there are some notable exceptions:

- For the maximum number of system allocated cores time series, the RICC workload presents no seasonal periods.
- For the maximum number of requested cores per hour time series, periods are mainly daily and weekly, but the Ciemat Euler workload present a daily period only.
- For the work volume per hour time series, the period is mainly daily, and weekly for ANL Intrepid workload, with the exception of the Filtered Metacentrum workload that shows no seasonal patterns.
- For the number of submitted jobs per hour, PIK IPLEX workload have no seasonal patterns. For other workloads, the daily period is the strongest seasonal pattern. Note that ANL and RICC workloads have equally strong weekly seasonal periods.

Table 1: Power spectral density values of different workload time series depending on daily (24h), semi-weekly (84h), weekly (168h), monthly (730h) and bi-monthly (1460h) seasonal period. "NoS" means that no seasonality has been detected.

Period	System allocated cores						
	ANL I.	Ciemat E.	Metac.	F. Metac.	PIK I.	RICC	UniLu G.
24h	2.7×10^{11}	1.7×10^{8}	3.0×10^{8}	NoS	3.8×10^{8}	NoS	NoS
84h	4.1×10^{11}	NoS	NoS	NoS	NoS	NoS	NoS
168h	$\mathbf{9.2 \times 10^{11}}$	$\mathbf{3.2 \times 10^{8}}$	3.3×10^{8}	$\mathbf{1.2 \times 10^{8}}$	4.3×10^{8}	NoS	$\mathbf{3.0 \times 10^{7}}$
730h	NoS	NoS	$\mathbf{4.2 \times 10^{8}}$	$\mathbf{1.5 \times 10^{8}}$	NoS	NoS	NoS
1460h	NoS	NoS	NoS	NoS	$\mathbf{5.7 \times 10^{8}}$	NoS	NoS
	Requested cores						
	ANL I.	Ciemat E.	Metac.	F. Metac.	PIK I.	RICC	UniLu G.
24h	$\mathbf{4.1 \times 10^{12}}$	$\mathbf{1.7 \times 10^{8}}$	$\mathbf{2.2 \times 10^{9}}$	$\mathbf{3.6 \times 10^{8}}$	$\mathbf{6.9 \times 10^{8}}$	$\mathbf{3.0 \times 10^{9}}$	1.4×10^{7}
168h	2.8×10^{12}	NoS	1.2×10^{9}	$\mathbf{4.5 \times 10^{8}}$	$\mathbf{6.2 \times 10^{8}}$	$\mathbf{3.6 \times 10^{9}}$	$\mathbf{3.8 \times 10^{7}}$
730h	NoS	NoS	9.2×10^{8}	1.9×10^{8}	NoS	NoS	NoS
	Work volume						
	ANL I.	Ciemat E.	Metac.	F. Metac.	PIK I.	RICC	UniLu G.
24h	$\mathbf{4.8 \times 10^{19}}$	$\mathbf{2.8 \times 10^{16}}$	$\mathbf{1.4 \times 10^{17}}$	NoS	$\mathbf{7.9 \times 10^{16}}$	$\mathbf{2.2 \times 10^{17}}$	$\mathbf{6.5 \times 10^{15}}$
168h	2.7×10^{19}	NoS	NoS	NoS	NoS	NoS	NoS
	Number of submitted jobs						
24h	$\mathbf{2.2 \times 10^{4}}$	$\mathbf{8.5 \times 10^{7}}$	$\mathbf{1.8 \times 10^{8}}$	$\mathbf{3.9 \times 10^{7}}$	NoS	$\mathbf{3.5 \times 10^{6}}$	$\mathbf{4.3 \times 10^{5}}$
168h	$\mathbf{2.5 \times 10^{4}}$	NoS	NoS	NoS	NoS	$\mathbf{3.1 \times 10^{6}}$	NoS
730h	NoS	4.5×10^{7}	NoS	NoS	NoS	NoS	NoS
1460h	NoS	4.3×10^{7}	NoS	NoS	NoS	NoS	NoS

In the case of the Wikimedia's workload, both time series are stationary by removing the trend. Periodograms show in both cases a very strong daily seasonality and less pronounced weekly and semi-weekly seasonality. In comparison with [13], an additional semi-weekly seasonality has been identified.

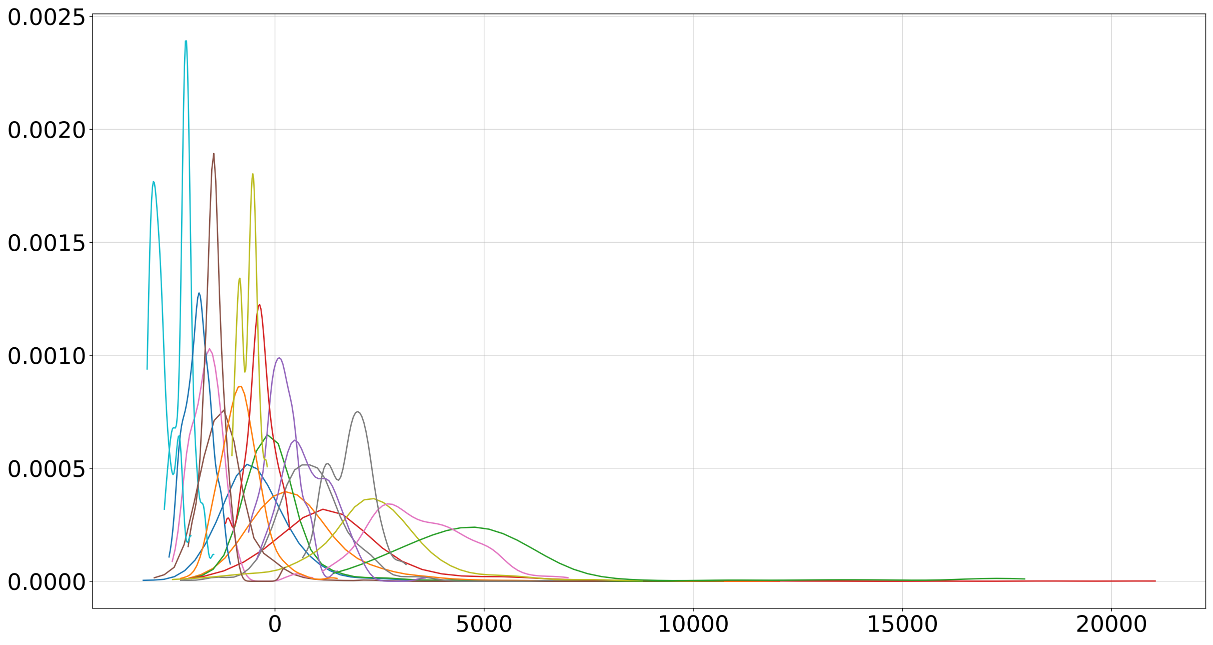
6.2 Seasonal distribution-based clustering

Having identified the different seasonality present in workloads, we now investigate their potential irregularities. In this section, we focus on determining if all samples of a seasonal period are based on the same distribution by using the distribution-based clustering algorithm presented in section 4.

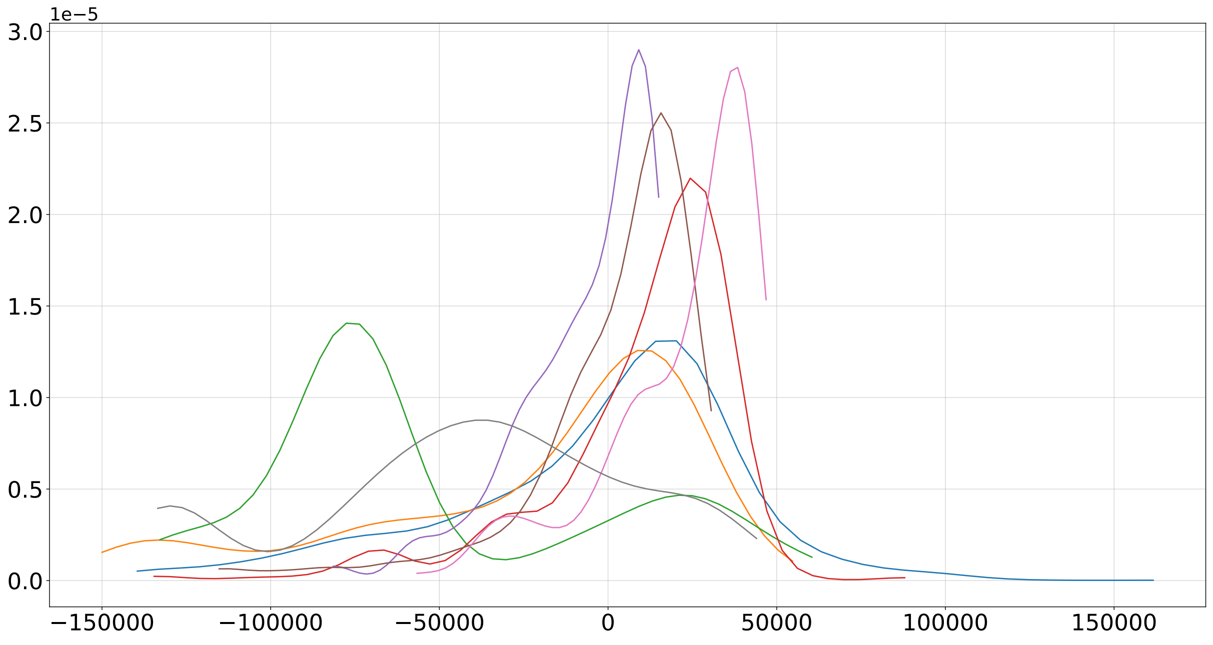
Fig. **??** shows some examples of clustering results of different workloads and time series for different time periods using the distribution-based clustering algorithm. Each line represents a cluster from which the samples in that cluster are derived: it is constructed using a smoothing process from the density distribution of the values of a time series. The number of clusters is given in Table 2 (presented after). In most cases, the difference between two clusters resides in the distribution tail from which the samples are derived (see the distribution tail in Fig. **??**), in some values being more present for certain distributions than others (see Fig. **??** and Fig. **??**), or in medians differing significantly when distributions of the clusters have the same shape (see Fig. **??**).

Table 2 shows the number of seasonal pattern samples and the number of clusters for the different time series studied. The maximum number of requested cores and system allocated cores time series are the most irregular in terms of distribution, considering the number of clusters. For example, the 3079 Ciemat Euler day samples are divided into 87 and 73 clusters, respectively, while there are only 13 clusters with the work volume time series. Note that there is an exception with the number of submitted jobs time series, which presents 233 clusters. In all cases, whatever the workload and time series, there are sufficient clusters to suggest that seasonality is irregular in terms of distribution. In other words, from one seasonal pattern to the next, the amplitude of the peak and/or trough may strongly vary.

In the case of the Wikimedia's workload, the request flow time series shows 3 clusters for daily seasonal pattern (with 365 samples), 4 clusters for semi-weekly seasonal pattern (with 104 samples) and 4 clusters for weekly seasonal pattern (with 52 samples). The number of bytes per hour time series presents approximately the same results with 4 clusters for daily seasonal pattern, 3 clusters for semi-weekly seasonal pattern and 4 clusters for weekly seasonal pattern. This shows that time series from this cloud workload are much less irregular than those from HPC workloads.



(a) Metacentrum - Number of requested cores - Daily periods



(b) ANL Intrepid - Number of system allocated cores - Semi-weekly periods

Figure 3: Distribution-based clustering results of different workload time series and for different time periods.

Table 2: Number of seasonal pattern samples / number of clusters for different workload time series depending on daily (24h), semi-weekly (84h), weekly (168h), monthly (730h) and bi-monthly (1460h) seasonal period. "NoS" means that no seasonality has been detected.

Period	System allocated cores						
	ANL I.	Ciemat E.	Metac.	F. Metac.	PIK I.	RICC	UniLu G.
24h	186 / 14	3079 / 87	416 / 22	NoS	1083 / 104	NoS	NoS
84h	53 / 8	NoS	NoS	NoS	NoS	NoS	NoS
168h	26 / 6	439 / 34	59 / 13	56 / 11	154 / 26	NoS	6 / 5
730h	NoS	NoS	13 / 6	13 / 4	NoS	NoS	NoS
1460h	NoS	NoS	NoS	NoS	17 / 9	NoS	NoS
	Requested cores						
	ANL I.	Ciemat E.	Metac.	F. Metac.	PIK I.	RICC	UniLu G.
24h	186 / 9	3079 / 73	416 / 20	397 / 21	1083 / 101	49 / 11	47 / 13
168h	26 / 5	NoS	59 / 9	56 / 12	154 / 26	7 / 3	6 / 5
730h	NoS	NoS	13 / 5	13 / 4	NoS	NoS	NoS
	Work volume						
	ANL I.	Ciemat E.	Metac.	F. Metac.	PIK I.	RICC	UniLu G.
24h	186 / 6	3079 / 16	416 / 4	NoS	1083 / 11	49 / 4	47 / 4
168h	26 / 6	NoS	NoS	NoS	NoS	NoS	NoS
	Number of submitted jobs						
	ANL I.	Ciemat E.	Metac.	F. Metac.	PIK I.	RICC	UniLu G.
24h	186 / 8	3079 / 233	416 / 28	397 / 20	NoS	49 / 5	47 / 2
168h	26 / 5	NoS	NoS	NoS	NoS	7 / 2	NoS
730h	NoS	101 / 29	NoS	NoS	NoS	NoS	NoS
1460h	NoS	50 / 21	NoS	NoS	NoS	NoS	NoS

6.3 Seasonal peaks

Finally, in this section, we investigate the appearance of seasonality peaks to assess their recurrence and considering the results given by the based-distribution clustering.

Fig. **??** shows frequency of occurrence of seasonal peaks of different workload time series and for different time periods. Note that, for each figure, only workloads and time series for which the seasonality in question has been detected are considered. In the case of daily periods (see Fig. **??**), peaks appear mainly between 9 am and 6 pm, and more rarely at night between 00 am and 7 am. This trend is similar for all workloads and all time series. For weekly periods (see Fig. **??**), seasonal peaks occur mainly during the day on weekdays.

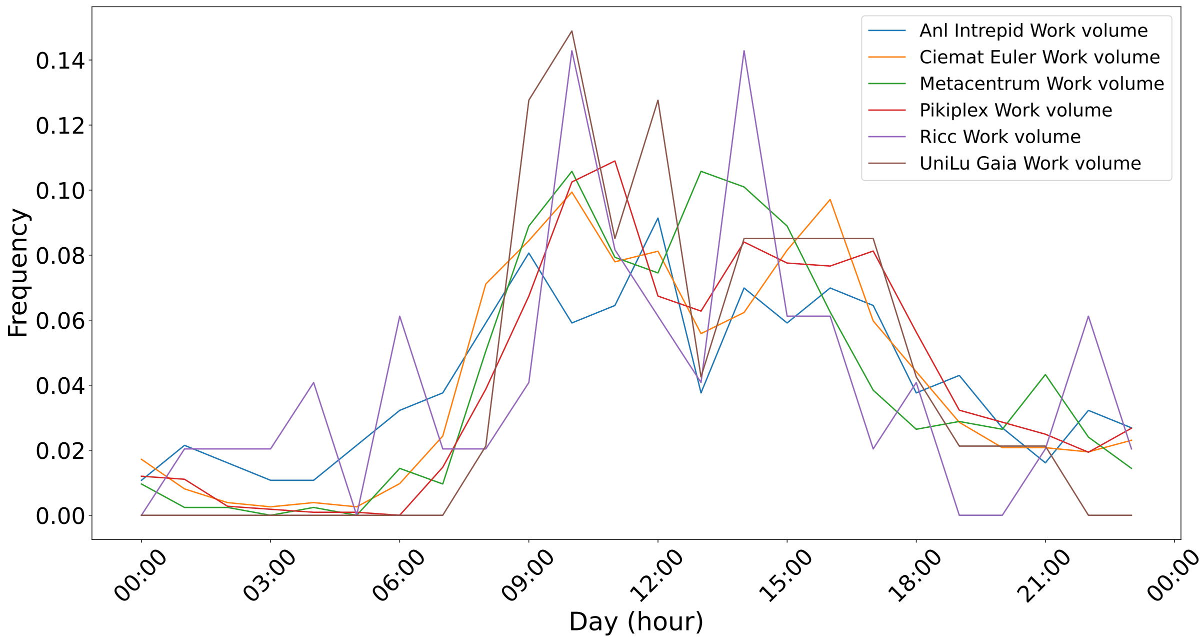
In the case of the Wikimedia's workload, seasonal peaks of the request flow time series mainly appear at 3 pm in 46% of cases, and between 1 pm and 11 pm in all cases (>99%), Monday in 60% of cases and Tuesday in 29% of cases. Seasonal peaks of the number of bytes per hour time series mainly appear also at 3 pm (40%) and between 1pm and 11pm (>99%), Monday (56%) and Tuesday (27%).

6.4 Synthesis

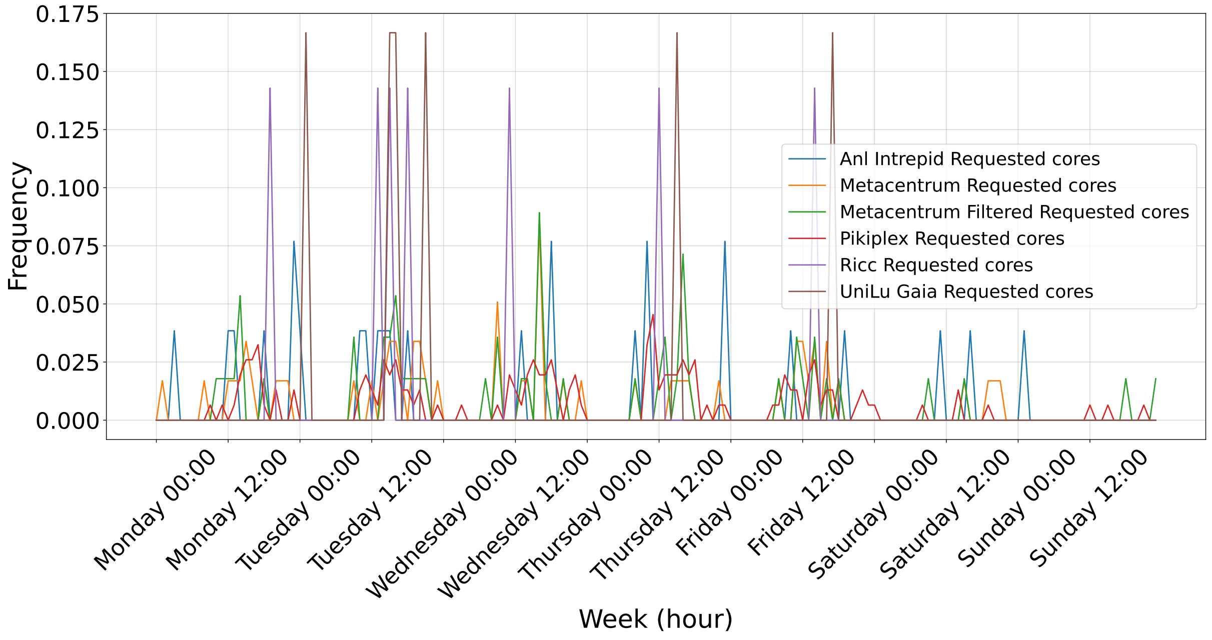
Box-plots can highlight seasonality, but are not a sufficient method due to the loss of dependency between successive values. Automatic time series decomposition is effective when there is only evident seasonality. Finally, ACFs are difficult to interpret, and the seasonal pattern(s) must be evident to be perceived. For these reasons, we propose to use an FFT technique (periodogram) that provides trivially clear evidence of the seasonal pattern(s). This technique requires the time series studied using the KPSS and ADF tests to be stationary.

Irregularities were evaluated using two methods: (i) a distribution-based clustering using the Kruskal-Wallis test, highlighting the number of distributions representing the samples of the seasonal pattern(s) of the time series, and (ii) an analysis of the frequency of occurrence of the seasonal peaks, indicating that peaks do not appear at the same hour or on the same day. We show that the studied time series for forecasting are not regular, and some of them are less regular from the distribution-based clustering algorithm point of view.

The study can be extended by studying the days composing the clusters (predominantly weekend days in some clusters, predominantly weekdays in others) and their size, to deduce the most dominant distributions. The same can be done with seasonal peaks, by separating weekdays and weekend days. If there are significant differences between clusters and/or between weekdays and weekend days, this could be used to switch to a different forecasting method, or to adapt the parameters of the method depending on the day.



(a) Work volume - Daily periods



(b) Number of requested cores - Weekly periods

Figure 4: Frequency of occurrence of seasonal peaks of different workloads and time series and for different time periods.

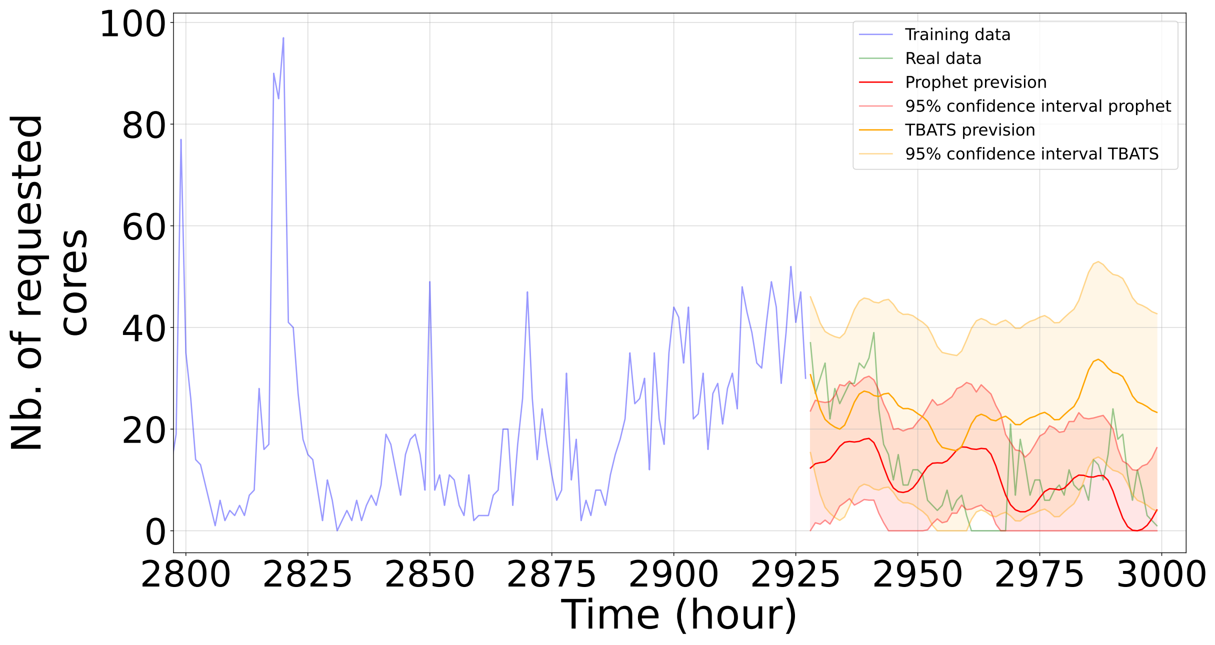


Figure 5: ANL Intrepid - Number of submitted jobs prevision using TBATS (orange) and Prophet (red).

Yearly seasonality was studied, particularly with the Ciemat Euler workload which has sufficient patterns (9 years) after filtering and considering the minimum number of patterns required to study seasonality according to [17], but this seasonality was not detected.

7 Characterization use case example

To illustrate the results of the previous sections, we apply the seasonal study using 2 multi-seasonal forecasting methods (TBATS and Prophet) to the number of submitted jobs time series from ANL Intrepid workload, which presents equally strong daily and weekly seasonal patterns identified using the periodogram. These seasonal patterns are input parameters for TBATS and Prophet. A total of 3000 observations divided into training data (2928) and test data (72) are considered.

Fig. **??** shows previsions using TBATS and Prophet forecasting methods, respectively. The prevision covers the next 72 hours. As can be seen, the 95% confidence interval covers the majority of real data, which shows the relevance of our prevision. TBATS confidence interval is larger than that of Prophet. The mean absolute error (MAE) and the root mean squared error (RMSE) made with TBATS and Prophet are 13.4 and 14.8, and 12.1 and 15.0, respectively. Prophet makes fewer errors than TBATS according to MAE, but they are greater according to RMSE. Note however that, due to the irregularity of seasonal patterns, particularly the appearance of seasonal peaks, previsions underestimate and overestimate the number of submitted jobs and, at some time steps, seasonal peaks appear before or after what the forecasting model predicts.

8 Conclusion

To study seasonal patterns of cloud workload time series, most of the literature use simple methods such as box-plots, automatic decomposition or ACFs, but these methods have their weak points. Being designed to detect periodicity in behaviors, FFT techniques are suitable to detect one or more strong or weak seasonal patterns. Moreover, these studies are limited to one or two time series and, in most cases, cloud workloads do not last enough time to confirm seasonal patterns, which can present some difficulties for generalization. For these reasons, we explore the seasonality of different time series from several HPC workloads in detail.

Seasonal patterns in HPC workloads are not obvious because of the many irregularities due to unexpected, delayed and/or extended user demands or IT system usage. In this work, we have studied the seasonality of different time series from several workloads to obtain information that will be useful when applying forecasting methods. We also proposed a distribution-based clustering algorithm using the Kruskal-Wallis test and a seasonal peak appearances analysis for assessing the irregularities of the time series. Results show that in most cases, several seasonality are present, but behave more or less irregularly.

These seasonality encourage us to use seasonality-based forecasting methods, such as ARMA, TBATS or Prophet. However, irregularities may reduce the performance of these methods. For this reason, in future work, we plan to compare with different seasonal forecasting methods, as well as other methods such as recurrent neural network (RNN) or support vector regression (SVR). The two forecasting models (IT system usage and user demand) will be assessed by investigating the most suitable forecasting methods for each time series. For the user demand forecasting model, forecasting methods will be combined in the best possible way to create different scenarios. Subsequently, we will integrate the results in a real environment (DATAZERO2).

Acknowledgments

We express our gratitude to Dror Feitelson for maintaining the Parallel Workloads Archive from which our input data come. Thanks also to Susan Coghlan and Narayan Desai for providing the ANL Intrepid workload log, Ciaron Linstead for PIK IPLEX, Motoyoshi Kurokawa for RICC, Joseph Emeras for UniLu Gaia, the Czech National Grid Infrastructure MetaCentrum for Metacentrum 2 and Antonio Juan Rubio-Montero for CIEMAT Euler.

References

[1] https://github.com/alibaba/clusterdata/, 2018. Accessed: 2024-01-30.

[2] K. Bandara, R. J. Hyndman, and C. Bergmeir. Mstl: A seasonal-trend decomposition algorithm for time series with multiple seasonal patterns. *arXiv preprint arXiv:2107.13462*, 2021.

[3] J. Bang, C. Kim, K. Wu, A. Sim, S. Byna, S. Kim, and H. Eom. Hpc workload characterization using feature selection and clustering. In *Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics*, pages 33–40, 2020.

[4] E. Betke and J. Kunkel. Footprinting parallel i/o–machine learning to classify application's i/o behavior. In *High Performance Computing: ISC Workshops 2019, Frankfurt, Germany, June 16-20, 2019, Revised Selected Papers 34*, pages 214–226. Springer, 2019.

[5] A. Burkimsher, I. Bate, and L. S. Indrusiak. A characterisation of the workload on an engineering design grid. In *Proceedings of the High Performance Computing Symposium*, pages 1–8, 2014.

[6] M. Calzarossa, L. Massari, and D. Tessera. Workload characterization: A survey revisited. *ACM Computing Surveys (CSUR)*, 48(3):1–43, 2016.

[7] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[8] W. J. Conover and R. L. Iman. Rank transformations as a bridge between parametric and nonparametric statistics. *The American Statistician*, 35(3):124–129, 1981.

[9] A. M. De Livera, R. J. Hyndman, and R. D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association*, 106(496):1513–1527, 2011.

[10] N. Dezhabad, S. Ganti, and G. Shoja. Cloud workload characterization and profiling for resource allocation. In *2019 IEEE 8th international conference on cloud networking (CloudNet)*, pages 1–4. IEEE, 2019.

[11] S. Di, D. Kondo, and W. Cirne. Characterization and comparison of cloud versus grid workloads. In *2012 IEEE International Conference on Cluster Computing*, pages 230–238. IEEE, 2012.

[12] D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979.

[13] A. A. Eldin, A. Rezaie, A. Mehta, S. Razroev, S. S. de Sjöstedt-de Luna, O. Seleznjev, J. Tordsson, and E. Elmroth. How will your workload look like in 6 years? analyzing wikimedia's workload. In *2014 IEEE Int. conference on cloud engineering*, pages 349–354. IEEE, 2014.

[14] D. Feitelson. *Workload modeling for computer systems performance evaluation.* Cambridge University Press, 2015.

[15] D. Feitelson, D. Tsafrir, and D. Krakov. Experience with using the parallel workloads archive. *Journal of Parallel and Distributed Computing*, 74(10):2967–2982, 2014.

[16] Hpc and htc tutorial. `https://www.grid5000.fr/w/HPC_and_HTC_tutorial`. Accessed: 2024-02-22.

[17] J. E. Hanke and D. W. Wichern. *Business forecasting.* Pearson Educación, 2005.

[18] A. S. Higginson, M. Dediu, O. Arsene, N. W. Paton, and S. M. Embury. Database workload capacity planning using time series analysis and machine learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 769–783, 2020.

[19] Z. Hou, H. Shen, X. Zhou, J. Gu, Y. Wang, and T. Zhao. Prediction of job characteristics for intelligent resource allocation in hpc systems: a survey and future directions. *Frontiers of Computer Science*, 16(5):165107, 2022.

[20] IEA. Data centres and energy – from global headlines to local headaches?, iea, paris. www.iea.org/commentaries/data-centres-and-energy-from-global-headline-to-local-headaches, 2019.

[21] S. Ilager, A. Toosi, M. Jha, I. Brandic, and R. Buyya. A data-driven analysis of a cloud data center: Statistical characterization of workload, energy and temperature.

[22] D. Klusáček, Š. Tóth, and G. Podolníková. Real-life experience with major reconfiguration of job scheduling system. In *Job Scheduling Strategies for Parallel Processing: 19th and 20th International Workshops, JSSPP 2015, Hyderabad, India, May 26, 2015 and JSSPP 2016, Chicago, IL, USA, May 27, 2016, Revised Selected Papers 19*, pages 83–101. Springer, 2017.

[23] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[24] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178, 1992.

[25] M. Masdari and A. Khoshnevis. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*, 23(4):2399–2424, 2020.

[26] A. Mishra, J. Hellerstein, W. Cirne, and C. Das. Towards characterizing cloud backend workloads: insights from google compute clusters. *ACM SIGMETRICS Performance Evaluation Review*, 37(4):34–41, 2010.

[27] H. Musbah, M. El-Hawary, and H. Aly. Identifying seasonality in time series by applying fast fourier transform. In *2019 IEEE Electrical Power and Energy Conference (EPEC)*, pages 1–4. IEEE, 2019.

[28] E. Patel and D. Kushwaha. Clustering cloud workloads: K-means vs gaussian mixture model. *Procedia computer science*, 171:158–167, 2020.

[29] J.-M. Pierson, G. Baudic, S. Caux, B. Celik, G. Da Costa, L. Grange, M. Haddad, J. Lecuivre, J.-M. Nicod, L. Philippe, V. Rehn-Sonigo, R. Roche, G. Rostirolla, A. Sayah, P. Stolf, M. Thi, and C. Varnier. Datazero: Datacenter with zero emission and robust management using renewable energy. *IEEE Access*, 7, 2019.

[30] G. Rodrigo, P.-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan. Towards understanding hpc users and systems: a nersc case study. *Journal of Parallel and Distributed Computing*, 111:206–221, 2018.

[31] M. Rodríguez-Pascual, A. J. Rubio-Montero, J. A. Moríñigo, and R. Mayo-García. Execution data logs of a supercomputer workload over its extended lifetime. *Data in brief*, 28:105006, 2020.

[32] A. A. Shahin. Using multiple seasonal holt-winters exponential smoothing to predict cloud resource provisioning. *arXiv preprint arXiv:1701.03296*, 2017.

[33] V. Shekhawat, A. Gautam, and A. Thakrar. Datacenter workload classification and characterization: An empirical approach. In *IEEE Int. Conf. on Industrial and Information Systems*, pages 1–7, 2018.

[34] K. Sindhu, K. Seshadri, and C. Kollengode. Workload characterization and synthesis for cloud using generative stochastic processes. *The Journal of Supercomputing*, 78(17):18825–18855, 2022.

[35] P. Singh, P. Gupta, and K. Jyoti. Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, 22(2):619–633, 2019.

[36] W. Tang, Z. Lan, N. Desai, D. Buettner, and Y. Yu. Reducing fragmentation on torus-connected supercomputers. In *2011 IEEE International Parallel & Distributed Processing Symp.*, pages 828–839. IEEE, 2011.

[37] S. J. Taylor and B. Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[38] M. Terai, R. Kashiwaki, and F. Shoji. Workload classification and performance analysis using job metrics in the k computer. 2017.

[39] The anl intrepid log. `https://www.cs.huji.ac.il/labs/parallel/workload/l_anl_int/index.html`. Accessed: 2024-02-22.

[40] The ciemat euler log. `https://www.cs.huji.ac.il/labs/parallel/workload/l_ciemat_euler/index.html`. Accessed: 2024-02-22.

[41] The metacentrum 2 log. `https://www.cs.huji.ac.il/labs/parallel/workload/l_metacentrum2/index.html`. Accessed: 2024-02-22.

[42] The periodogram. `https://online.stat.psu.edu/stat510/lesson/6/6.1`. Accessed: 2024-04-04.

[43] The potsdam institute for climate impact research (pik) ibm idataplex cluster log. `https://www.cs.huji.ac.il/labs/parallel/workload/l_pik_iplex/index.html`. Accessed: 2024-02-22.

[44] The ricc log. `https://www.cs.huji.ac.il/labs/parallel/workload/l_ricc/index.html`. Accessed: 2024-02-22.

[45] The university of luxemburg gaia cluster log. `https://www.cs.huji.ac.il/labs/parallel/workload/l_unilu_gaia/index.html`. Accessed: 2024-02-22.

[46] M. Thi, J. Pierson, G. D. Costa, P. Stolf, J. Nicod, G. Rostirolla, and M. Haddad. Negotiation game for joint IT and energy management in green datacenters. *Future Gener. Comput. Syst.*, 110:1116–1138, 2020.

[47] J. Wilkes. Yet more Google compute cluster trace data. Google research blog, Apr. 2020. Posted at `https://ai.googleblog.com/2020/04/yet-more-google-compute-cluster-trace.html`.

[48] T. Xie and J. Ding. Forecasting with multiple seasonality. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 240–245. IEEE, 2020.

[49] Z. Zhou, J. Sun, and G. Sun. Automated hpc workload generation combining statistical modeling and autoregressive analysis. In *International Symposium on Benchmarking, Measuring and Optimization*, pages 153–170. Springer, 2023.