Modeling and categorization of user trajectories in supercomputing systems using multi-aspect graphs

Arthur Henrique Craveiro Costa , Juan Humberto Leonardo Fabian , Antônio Tadeu Azevedo Gomes

[1]Laboratório Nacional de Computação Cientıífica (LNCC)
Petrópolis – RJ – Brasil

craveiro@posgrad.lncc.br, (juanhlf,atagomes)@lncc.br

Abstract. *Efficient resource management in supercomputing systems requires an accurate characterization of user behavior. Traditional workload characterization methods often fail to capture the multidimensional interactions between users and these systems. In this work, we propose a novel framework for modeling and categorizing user trajectories in supercomputers using Multi-Aspect Graphs (MAG). In our approach, each job is represented as a MAG vertex integrating five complementary aspects: requested queue, quantized CPU and memory utilization, user context, chronological submission order, and computational magnitude. Directed edges connect consecutive jobs, enriched with user identifiers and calculated "think times" to maintain authorship and temporal dynamics. To compare user behaviors, we adapt a sequence alignment algorithm for MAGs, combining aspect and temporal dissimilarity metrics. Subsequent trajectory clustering enables the identification of distinct, overlapping behavioral profiles. We evaluated our framework on a real-world job log, demonstrating its ability to preserve individual trajectory structures, reveal submission cadences, and group users with similar behavioral patterns without collapsing distinct dimensions into generic statistical aggregates.*

1. Introduction

Supercomputers are key to advancing modern scientific research, supporting complex simulations and data analysis. As these systems grow in scale and complexity, efficiently managing their resources becomes increasingly challenging. A critical component of resource usage optimization is the accurate characterization of user behavior. Traditionally, pieces of work in this area rely on job logs to extract metrics regarding resource allocation, execution time, and system utilization [Simakov et al. 2018].

Despite the availability of job logs, modeling user behavior remains complex because user-system interactions are multidimensional. Traditional workload characterization techniques, such as conventional graph models [Feitelson 2001] and isolated statistical analyzes [Schlagkamp et al. 2016], fail to represent in a unified way the structural context of the projects in which a user is involved, the explicit resource requests he/she makes, the actual computing utilization his/her jobs achieve, and the temporal sequence of his/her actions. Crucially, there is a need for a more expressive modeling framework capable of integrating these dimensions into cohesive user trajectories.

In this work, we propose a novel framework for modeling and categorizing user trajectories in supercomputing systems. We employ *Multi-Aspect Graphs* (MAG) [Wehmuth et al. 2016], a generalization of the traditional graph concept designed

to model relationships across time-varying or multilayered networks. In our approach, each job is defined as a vertex that combines five aspects: (i) the explicit intention of the user regarding the targeted queue; (ii) a quantized profile of the actual CPU and memory consumption of the job; (iii) the institutional and scientific context of the user; (iv) the chronological ordering of job submissions; and (v) a quantized representation of the job's overall computational magnitude. By linking consecutive jobs via directed edges enriched with user identification and calculated "think times", our MAG model preserves the authorship of transitions. To compare trajectories, we adapt a sequence alignment algorithm for MAGs, combining aspect and temporal dissimilarity metrics. Subsequent trajectory clustering enables the identification of distinct, overlapping groups of behavioral profiles. Overall, our framework provides a foundation for analyzing user trajectories individually or in groups, potentially allowing the extraction of deeper behavioral patterns than individual job statistics or aggregate workload distributions.

We evaluated our framework on six months of job logs from the Santos Dumont supercomputer,[1] comprising 83 109 jobs from 285 users. Results revealed distinct user profiles and recurring patterns in job submissions, resource utilization, and transitions between development and production queues. These findings illustrate the framework's potential to characterize user behavior and give additional context to workload analyses.

The remainder of this paper is organized as follows: Section 2 comments on related work; Section 3 describes the proposed methodology; Section 4 assesses the potential of the methodology with a real use case; Section 5 presents some concluding remarks and topics for future work.

2. Related work

The related literature can be organized into three lines: (i) workload characterization, (ii) user-centered behavioral analysis, and (iii) graph-based representations of HPC data.

In workload characterization, studies based on job log analysis jointly examined job size, runtime, memory, queues, inter-arrival times, and differences across scientific fields [You and Zhang 2013]. Long-term analyzes revealed persistent regularities and changes in job scale, waiting time, and resource requests [Patel et al. 2020], while user-level studies identified recurring submission characteristics and groups of related jobs [Wang et al. 2021]. Although these pieces of work demonstrate workload heterogeneity and recurring user patterns, they rely on distributions, correlations, or aggregated profiles rather than jointly representing the chronological, contextual, and resource dimensions of each submission.

In user-centered analysis, studies related job response time and computational complexity to the delay before a subsequent submission (*think time*), while acknowledging limitations when jobs execute concurrently [Schlagkamp et al. 2016]. Studies based on workload replay incorporated feedback so that submission behavior adapts to simulated system performance [Madon et al. 2024]. Other studies addressed resource-request accuracy [Jakobsche et al. 2023], combining user, project, and file-system metadata to characterize storage behavior [Lim et al. 2017], or examining responses to scheduling-policy changes [dos Santos and Gomes 2024]. These pieces of work show that job logs

[1] `https://sdumont.lncc.br`

contain behavioral information beyond aggregate load, but generally isolate the involved mechanisms instead of preserving them as a unified user trajectory.

Finally, graph-based approaches provide richer relational representations, but usually pursue other objectives. In these approaches, correlated job submissions were used to infer social communities [Zheng et al. 2011], or to support predictions of job outcomes and resource requirements [Bose et al. 2021]. As such, their vertices and edges represent social relations or predictive features rather than the ordered multidimensional states traversed by a user. Outside the HPC domain, patient pathways have been modeled as Multi-Aspect Graphs and compared using temporal dissimilarity [Rosa et al. 2023], providing a methodological precedent for the present work. Within the reviewed HPC literature, however, we found no approach that combines resource requests, observed utilization, project context, chronological order, computational magnitude, and intervals between submissions to describe and categorize user behavior in supercomputing systems.

3. Methodology

3.1. User Trajectories as a Multi-Aspect Graph

The Multi-Aspect Graph (MAG) [Wehmuth et al. 2016] generalizes the graph concept to model time-varying or multilayer relationships between objects. A traditional graph is defined as $\mathbb{G} = (V, E)$, where V is a set of *nodes* (representing indivisible objects) and $E \subseteq V \times V$ is a set of *edges*. In contrast, a MAG $\mathbb{H}$ is defined as $\mathbb{H} = (X, E)$, where $X = [X_1, \cdots, X_n]$ is a finite list of finite sets, each of which is called an *aspect*. Thus, instead of nodes that typify indivisible entities, the nodes of a MAG are n-tuples whose length n is the number of aspects of the MAG. Hence, the edges in E are tuples of length $2n$; the first n elements correspond to the origin node, while the last n elements correspond to the target node. Each aspect represents a specific object feature in the graph.

In our modeling domain, the definition of the aspects aimed to capture complementary dimensions of user-system interaction, combining information on resource requests, resource utilization, request context, temporal order, and computational magnitude of each job. A job is then represented as a MAG vertex composed of five aspects: $(Request, Utilization, Project, Sequence, Geometry)$. The identity of each vertex is determined by the complete combination of its aspect values, with the $Sequence$ aspect preserving the distinction between successive job occurrences in the full MAG.

Figure 1 exemplifies how a fictitious job log from a user is used to build a MAG representing his/her trajectory. Figure 1b is the result of an ETL procedure that extracts from the actual job log in Figure 1a the information needed to build the MAG depicted in Figure 1c. Since each aspect in a MAG is a finite set, but some data in an actual job log are continuous values (e.g. submission time, elapsed time, amount of consumed memory), quantization strategies need to be employed for some of the aspects. The different elements appearing in this example are detailed in the remainder of this subsection.

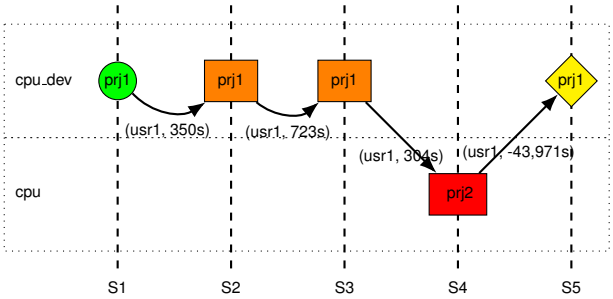
Without loss of generality, we use the format of job logs recorded by the Slurm job scheduler [SchedMD 2023] to describe how the five aspects and the edges in the resulting MAG are built in the following paragraphs. We select 12 fields (out of 107) from Slurm job logs, grouping them into: (i) identification (`JobID`, `User`, and `Account`); (ii) resources (`AllocCPUS`, `NTasks`, and `AllocTRES`); (iii) metrics (`Submit`, `End`, `ElapsedRaw`, `TotalCPU`, and `MaxRSS`); and (iv) system (`Partition`).

JobID	User	Account	AllocCPUS	NTasks	AllocTRES	Submit	End	ElapsedRaw	TotalCPU	MaxRSS	Partition
10015	usr1	prj1	12	1	mem=32Gb	07:30:01	07:45:10	...	...	...	cpu_dev
10031	usr1	prj1	24	2	mem=64Gb	07:51:00	08:10:07	...	...	...	cpu_dev
10097	usr1	prj1	24	2	mem=64Gb	08:22:10	08:50:09	...	...	...	cpu_dev
10135	usr1	prj2	96	8	mem=256Gb	08:55:13	21:13:11	...	...	...	cpu
10172	usr1	prj1	12	1	mem=32Gb	09:00:20	09:18:12	...	...	...	cpu_dev

(a) Example job log (Slurm fields)

Sub.	End	Req.	Utl.	Prj.	Geo.
07:30:01	07:45:10	cpu_dev	(1,1)	prj1	SR
07:51:00	08:10:07	cpu_dev	(2,2)	prj1	LR
08:22:10	08:50:09	cpu_dev	(2,2)	prj1	LR
08:55:13	21:13:11	cpu	(2,2)	prj2	LE
09:00:20	09:18:12	cpu_dev	(2,1)	prj1	Md

(b) Example job log (post-processed)



(c) Pictorial representation of the MAG

Figure 1. Example of a single user trajectory as a MAG. In (c) the geometric shape of each vertex represents the aspect $Utilization$ (Utl.); its color indicates the aspect $Geometry$ (Geo.); their labels indicate the aspect $Project$ (Prj.); its horizontal position (following the dashed vertical lines) indicates the aspect $Sequence$, obtained from the job submission time (Sub.); and the "lanes" where each node is located represent the aspect $Request$ (Req.). Each edge has attributes identifying a user and a think time.

Request. This aspect represents the queue or partition requested by the user and was mapped directly from the `Partition` field. This aspect reflects the user's explicit intention regarding the type of computational resource required, such as CPU-focused partitions, GPU partitions, shared nodes, long-allocation nodes, or large memory nodes.

Utilization. This aspect characterizes the level of resource usage by the job. It is important to note that, in this work, the aspect $Utilization$ does not incorporate specific GPU utilization metrics. Although partitions associated with GPUs are preserved in the aspect $Request$, the occupancy profile was defined exclusively from host CPU time and RAM memory. This choice stems from the lack of effective GPU usage metrics in the job logs used for this work (see Section 4). Thus, even for jobs submitted to GPU-enabled queues, the computed utilization should be interpreted as CPU and memory utilization, while the presence of the GPU remains represented by the requested queue/partition.

This aspect was built from two utilization measures:

(i) **CPU Utilization** (U_{cpu}), the ratio between the CPU time effectively used and the maximum allocated processing potential:

$$U_{cpu} = \frac{\texttt{TotalCPU}}{\texttt{ElapsedRaw} \times \texttt{AllocCPUS}} \times 100$$

(ii) **Memory Utilization** (U_{mem}), the ratio between peak resident memory and total allocated memory. To account for the parallelism recorded by the job scheduler, we take the memory peak (`MaxRSS` field) times the number of parallel tasks in the job:

$$U_{mem} = \frac{\texttt{MaxRSS} \times \texttt{NTasks}}{\texttt{AllocTRES}_{mem}} \times 100$$

Finally, both U_{cpu} and U_{mem} are quantized into quartiles (1 to 4). The aspect $Utilization$ is defined as the concatenation of the CPU quartile and the memory quartile. For example, "(1,4)" indicates low CPU utilization and high memory utilization. This results in 16 possible states that characterize the resource consumption profiles of the job.

Project. This aspect represents the institutional and scientific context of the job allocation (e.g. "Computational Materials Physics" or "Water Study at Electrochemical Interfaces"). It is extracted directly from the `Account` field. This aspect allows for differentiating trajectories associated with distinct projects and preserving in MAG the fact that the same user can submit jobs in different research contexts.

Sequence. This aspect captures the overall chronological order of job submissions. It is obtained from the ordering of jobs by the `Submit` field. Besides preserving the order of events, this aspect also differentiates occurrences that could share the same values in the other 4 aspects; therefore, it is a structural component of the trajectory, not a category that expresses computational characteristics of the jobs.

Geometry. This aspect synthesizes the "computational size" of the job, combining the amount of computing resources (number of cores) and the duration of their allocation. The base metric is the product:

$$CS = \texttt{AllocCPUS} \times \texttt{ElapsedRaw}$$

In supercomputers, load metrics often exhibit strongly heavy-tailed distributions, covering multiple orders of magnitude [Feitelson 2001]. For this reason, we apply logarithmic quantization to CS. For the job logs used in this study, we compared three quantization strategies over $\log_{10}(CS)$: (i) K-Means [MacQueen 1967]; (ii) quantile-based discretization (similar to the aspect $Utilization$); and (iii) hierarchical Ward clustering [Ward 1963], with sparse connectivity between adjacent observations in the logarithmic scale. For each strategy, configurations ranging from three to eight classes were evaluated using three validation metrics: (i) the silhouette score [Rousseeuw 1987]; (ii) the Davies-Bouldin index [Davies and Bouldin 1979]; and (iii) the Calinski-Harabasz index [Caliński and Harabasz 1974]. Based on this evaluation, we selected the quantization obtained by K-Means with $k = 6$ classes, because it presented the highest silhouette score while providing six interpretable categories and a lower computational cost than Ward clustering. The class boundaries were defined by the midpoints between adjacent centroids in the logarithmic scale. The six final categories were labeled ordinally to reflect a progression of computational magnitude: *Small-Mini* (SM), *Small-Tiny* (ST), *Small-Regular* (SR), *Medium* (Md), *Large-Regular* (LR), and *Large-Extra* (LE).

Edges. Creating edges in the MAG follows the logic of individual users' trajectories. For each user, the jobs are ordered by the aspect $Sequence$. Next, a directed edge is created connecting the vertices of consecutive jobs (Job_i and Job_{i+1}):

$$E_i = (R_i, U_i, P_i, S_i, G_i) \rightarrow (R_{i+1}, U_{i+1}, P_{i+1}, S_{i+1}, G_{i+1})$$

Each edge is enriched with two attributes:

(i) **User:** user identifier, allowing the complete graph to store multiple trajectories simultaneously. This attribute is necessary because different users may traverse the same composite states or perform structurally similar transitions; therefore, user identification preserves the authorship of the transition and allows for the reconstruction of individual trajectories from the aggregated MAG.

(ii) **Think Time:** time interval between the end of the execution of Job_i and the submission of Job_{i+1}:

$$\textit{think_time}_i = \texttt{Submit}_{i+1} - \texttt{End}_i$$

This attribute supports behavioral analysis by indicating patterns compatible with automated (batch) submissions or interactive work sessions, although it does not determine the submission mechanism by itself. Negative values indicate temporal overlap between jobs (e.g. concurrent submissions).

3.2. Dissimilarity measure

User trajectories are ordered sequences of states and transitions in the MAG, where each state corresponds to the characteristics associated with a job. Since these characteristics are represented by the aspects of the MAG, a state is formalized by a composite vertex $v_i \in \mathcal{V}(H)$. For a user u with n ordered jobs, their trajectory can be represented by

$$\mathcal{T}_u = (V_u, \Delta_u), \qquad V_u = (v_1, v_2, \ldots, v_n), \qquad \Delta_u = (\tau_1, \tau_2, \ldots, \tau_{n-1}), \tag{1}$$

where V_u is the sequence of composite vertices visited by the user and Δ_u comprises the time intervals associated with transitions between consecutive states. Thus, each pair (v_i, v_{i+1}) corresponds to a directed edge in the MAG and represents the transition from the state associated with job_i to the state associated with the subsequent job. The interval τ_i complements this relationship with information regarding the time separation between the two events, which is defined as the *think time*. The term *trajectory* should not be confused with the concept of *path* in graph theory.

The purpose of trajectory comparison is to measure the behavioral similarity between two sequences. A direct correspondence between positions at the same index is rarely appropriate: users may have varying numbers of jobs, initiate equivalent activities at different positions, and present additional or missing events. Furthermore, each state comprises multiple aspects, and structurally similar transitions may occur with differing time intervals. Thus, comparing trajectories requires simultaneously considering state content, the relative order of events, and the temporal dimension. A *dissimilarity* measure is then defined to compare user trajectories.

To quantify the dissimilarity between two trajectories, an approach based on global sequence alignment is adopted, inspired by the Needleman-Wunsch algorithm [Needleman and Wunsch 1970] and the adaptation proposed by [Rosa et al. 2023]. In this work, the operation analogous to substitution corresponds to the alignment between events described by multi-aspect vertices, while insertions and removals are represented by event-skipping operations. The alignment cost incorporates a content component, based on the MAG aspects, and a temporal component, based on think time.

Adopting the representation defined in Equation (1), let $\mathcal{T}^A = (V^A, \Delta^A)$ and $\mathcal{T}^B = (V^B, \Delta^B)$ be the trajectories of users A and B. Here, $V^A = (v_1^A, \ldots, v_n^A)$ and

$V^B = (v_1^B, \ldots, v_m^B)$ are the sequences of composite vertices, while Δ^A and Δ^B comprise the sequences of intervals between consecutive events. As mentioned, these intervals may take negative values, meaning a temporal overlap of jobs. The objective is to obtain an optimal correspondence between events from the two trajectories—allowing for alignments and skips—that minimizes a total cost composed of aspect and temporal dissimilarities.

Aspect Dissimilarity. The aspect distance, denoted by $distA$, quantifies the content dissimilarity between two composite vertices $v_i \in V^A$ and $v_j \in V^B$. Each vertex is represented by the 5-tuple $v = (R, U, P, S, G)$, corresponding to the aspects *Request*, *Utilization*, *Project*, *Sequence*, and *Geometry*. The aggregated distance is defined as a weighted sum of the partial distances for each aspect (Equation 2):

$$distA(v_i, v_j) = \sum_{k \in \{R,U,P,S,G\}} w_k \cdot dist_k(v_i, v_j) \tag{2}$$

The partial distances $dist_k$ were defined according to the nature of each aspect, with weights determined empirically to reflect the relative importance of these aspects in characterizing HPC usage behavior. Thus, the measure prioritizes aspects related to resource occupancy and the computational scale of the jobs, without entirely disregarding the context of the request and the project.

- **Utilization ($w_U = 0.45$):** represents the combined class of CPU and memory efficiency. Since each category consists of two quartiles, $U = (q_{cpu}, q_{mem})$, with $q_{cpu}, q_{mem} \in \{1, 2, 3, 4\}$, the distance uses the normalized Manhattan metric:

$$dist_U(U_i, U_j) = \frac{|q_{cpu}^i - q_{cpu}^j| + |q_{mem}^i - q_{mem}^j|}{6}. \tag{3}$$

  The distance is normalized by 6, which reflects the maximum potential distance between the usage categories $(1, 1)$ and $(4, 4)$.
- **Geometry ($w_G = 0.30$):** represents the job's computational scale class, derived from `cpu_seconds`. Since the final categories are discrete, a binary distance is used, where $dist_G = 0$ when the categories match and $dist_G = 1$ otherwise.
- **Request ($w_R = 0.15$):** uses a semantic distance between partitions, based on the proximity between requested resource types. It distinguishes three conceptual situations: equality between partitions ($dist_R = 0$); functional proximity, where partitions belong to the same resource family or represent equivalent variants regarding development, sharing, long duration, or increased memory ($0 < dist_R < 1$); and the absence of a direct relationship ($dist_R = 1$). Thus, intermediate values represent degrees of proximity between queues, while the maximum distance is reserved for requests expressing substantially different usage intentions.
- **Project ($w_P = 0.10$):** considers the thematic proximity between projects. Again, there are three conceptual situations: same project ($dist_P = 0$); distinct projects belonging to the same research group ($0 < dist_P < 1$); projects belonging to different groups ($dist_P = 1$). This aspect preserves the scientific context of the submission without allowing it to dominate the final measure.
- **Sequence ($w_S = 0.00$):** does not contribute directly to $distA$, as the absolute ordinal position should not induce similarity between distinct users. The order of events is incorporated by the alignment algorithm itself, which operates progressively on initial segments of the trajectories.

Temporal Dissimilarity. The temporal component aims to capture differences in submission cadence between users. Let t_1 and t_2 be the accumulated times since the last aligned pair, measured by the sum of the think times elapsed between events that have not yet been aligned. Thus, when the algorithm skips events in one of the sequences, the corresponding accumulated time is incremented. When an alignment occurs, the temporal accumulators are reset. This mechanism allows the temporal cost to reflect session gaps and misalignments, rather than simply comparing local pairs of events.

Temporal dissimilarity is defined as:

$$distT(t_1, t_2) = \begin{cases} 1, & \text{if } (t_1 < 0 \leq t_2) \text{ or } (t_2 < 0 \leq t_1), \\ \min\left(\dfrac{|t_1 - t_2|}{\epsilon}, 1\right), & \text{otherwise.} \end{cases} \tag{4}$$

where the distance is normalized by a parameter ϵ, representing the upper limit of the temporal cost. A value of $\epsilon = 86\,400$ seconds (1 day) was adopted. Negative think time values, which indicate job overlap, were preserved because they represent a behavioral pattern distinct from a pause between submissions. For this reason, when one trajectory accumulates negative time and the other accumulates non-negative time, the temporal dissimilarity is considered maximal.

Dissimilarity Algorithm. The algorithm follows the dynamic programming formulation for alignment, where alignment becomes computationally feasible by exploiting two properties: the existence of overlapping subproblems and optimal substructure. The optimal alignment of two prefixes depends on solutions already obtained for smaller prefixes. Thus, for sequences of lengths n and m, a table with $(n+1) \times (m+1)$ positions is constructed, in which each cell records the best cost for aligning the corresponding prefixes. The complete solution is obtained by incrementally combining these partial solutions, avoiding the explicit enumeration of all possible alignments. The version adopted is iterative (*bottom-up*) and considers three operations: aligning events, skipping an event in A, or skipping an event in B.

The algorithm populates a cost matrix $D_{(n+1)\times(m+1)}$, where $D[i, j]$ stores the minimum cost to align the initial segments $(v_1^A, \ldots, v_i^A)$ and $(v_1^B, \ldots, v_j^B)$. The additional dimension represents the empty segment of each trajectory, which is necessary to initialize the base cases where all events in one of the sequences are skipped. In parallel, an auxiliary structure is maintained to track the accumulated times $(t_{i,j}^A, t_{i,j}^B)$ since the last alignment, which is required for calculating $distT$.

The alignment cost between (v_i^A, v_j^B) combines aspects and time, following the temporal adaptation of trajectory alignment proposed by [Rosa et al. 2023]:

$$c_{align}(i, j) = \omega_A \cdot distA(v_i^A, v_j^B) + (1 - \omega_A) \cdot distT(t_{i-1,j-1}^A + \tau_i^A,\, t_{i-1,j-1}^B + \tau_j^B) \tag{5}$$

where ω_A controls the trade-off between aspects and time. We set $\omega_A = 0.8$ (meaning 80% of the cost is determined by aspects and 20% by time).

To avoid semantically incoherent alignments, the *align* operation is permitted only when the pair of events is sufficiently similar in terms of aspects—that is, when

$distA(v_i^A, v_j^B) \leq \delta$, with $\delta = 0.5$. Otherwise, the algorithm is restricted to *skip* operations. The *skip* operations carry a fixed penalty cost of 1.0, controlling the impact of insertions/deletions on the total cost.

Adapting the temporal alignment recurrence proposed by [Rosa et al. 2023], the cost matrix is populated as follows:

$$D[i,j] = \min \begin{cases} D[i-1, j-1] + c_{align}(i,j), & \text{if } distA(v_i^A, v_j^B) \leq \delta \\ D[i-1, j] + penalty, & \text{(skip in A)} \\ D[i, j-1] + penalty & \text{(skip in B)} \end{cases} \tag{6}$$

This iterative approach was adopted over a recursive version with memoization due to stability considerations—particularly for long trajectories—and because it simplifies the application of the metric to multiple pairs of users.

Finally, the resulting cost $D[n, m]$ is normalized by the sum of the lengths of the two trajectories, yielding a measure within the range $[0, 1]$ that allows for comparison between users with histories of different lengths:

$$dissim_{norm}(A, B) = \frac{D[n, m]}{|V^A| + |V^B|} \tag{7}$$

Based on this measure, pairs of users can be selected using a dissimilarity threshold and converted into similarity relations in a subsequent clustering step.

3.3. Trajectories Clusterization

After quantifying the distances between user trajectories, the next step is to cluster these users based on the similarity of their behaviors. To this end, we employ a graph-based approach that explicitly preserves the relational structure of the calculated similarities. The aim of the clustering process is to identify groups of jobs that share similar usage patterns, enabling the characterization of typical usage profiles.

Similarity Graph. Let $\mathbb{G}_{sim} = (V, E)$ be an undirected, weighted graph in which each vertex $v \in V$ represents a user and each edge expresses a relationship of behavioral proximity between the trajectories of two users.

The set V encompasses all users with a minimum trajectory length. The graph edges are established based on the normalized dissimilarity ($dissim_{norm}$). To transform dissimilarity into a measure of affinity, the edge weight (similarity) is defined as:

$$sim_{uv} = 1.0 - dissim_{norm}(u, v) \tag{8}$$

Only pairs with $dissim_{norm} \leq 0.5$ (equivalently, $sim_{uv} \geq 0.5$) are connected. This criterion reduces noise by restricting the graph to relationships with minimum similarity, preventing weak connections from dominating the global structure.

Clustering Algorithm. We used the OSLOM (*Order Statistics Local Optimization Method*) algorithm [Lancichinetti et al. 2011] to identify clusters (*modules*, in OSLOM terminology) in the graph $\mathbb{G}_{sim}$. This choice is primarily based on the algorithm's ability

to detect overlapping modules, which, in the context of our problem, correspond to user trajectories associated with multiple groups. Consequently, it is not limited to optimizing a global partition, as it takes into account the statistical evidence of each module relative to the network structure. Only users incident to at least one edge are considered as input for the algorithm; thus, isolated users remain accounted for solely within the similarity graph. This distinction allows for separating the absence of similarity relationships in the graph from the formation of unitary modules.

As the algorithm incorporates stochastic components and its results depend on configuration parameterization, a grid search was conducted considering different values for the significance threshold t that governs the statistical criterion for module acceptance, the coverage parameter cp that influences the solution's coverage and granularity, and the number of runs r, as well as multiple seeds. For each configuration, stability across partitions is quantified using the adjusted Omega index [Collins and Dent 1988], which is suitable for comparing cluster structures that may overlap. The search employed the undirected, weighted version of OSLOM, while preserving *singletons*.

4. Results

The proof of concept used job logs from the Santos Dumont supercomputer between $01/01/2023$ and $30/06/2023$, a period chosen for its stable queue configuration. Various data processing steps were performed to remove less relevant information, such as obsolete partitions, educational projects, and records lacking complete metrics. A minimum user trajectory length of 10 was also established; based on this criterion, the MAG was constructed using 83 109 jobs from 285 users. Table 1 shows the cardinality of the finite sets defining the most important aspects in the resulting MAG.

Table 1. Cardinality of the MAG's aspects for the Santos Dumont supercomputer.

Aspect	Cardinality
Project	122
Request	24
Utilization	16
Geometry	6

Figure 2 shows the trajectory of a single user. The X-axis represents the aspect *Sequence*, indicating the chronological order of submitted jobs. The Y-axis represents the aspect *Request*, used to analyze queue/partition selection patterns. The "points" represent the aspects *Utilization* and *Geometry*, and the lines represent edges associated with successive transitions and corresponding think times. In this figure, we can: (i) identify a cadence pattern of predominantly short think times, (ii) recognize concurrent job submissions (think time < 0), and (iii) detect abrupt changes in request, utilization, and geometry patterns. More specifically, we can initially observe a steady increase in utilization and geometry in queue `sequana_cpu_dev` (a development queue, which allows allocation of a single node for 20 minutes) in jobs with $Sequence = 1, \ldots, 3$, followed by a change of queue at $Sequence = 4$ to `sequana_cpu` (a production queue, which allows allocation of up to 20 nodes for a couple of days). This change of behavior, possibly a result of exhausting resources and/or time in the former queue, occurs again in jobs with $Sequence = 6, \ldots, 8$ and then $Sequence = 9$, followed by a change to

GPU nodes (queues `sequana_gpu_dev` and `sequana_gpu`) in the subsequent jobs ($Sequence = 12, \ldots, 18$).

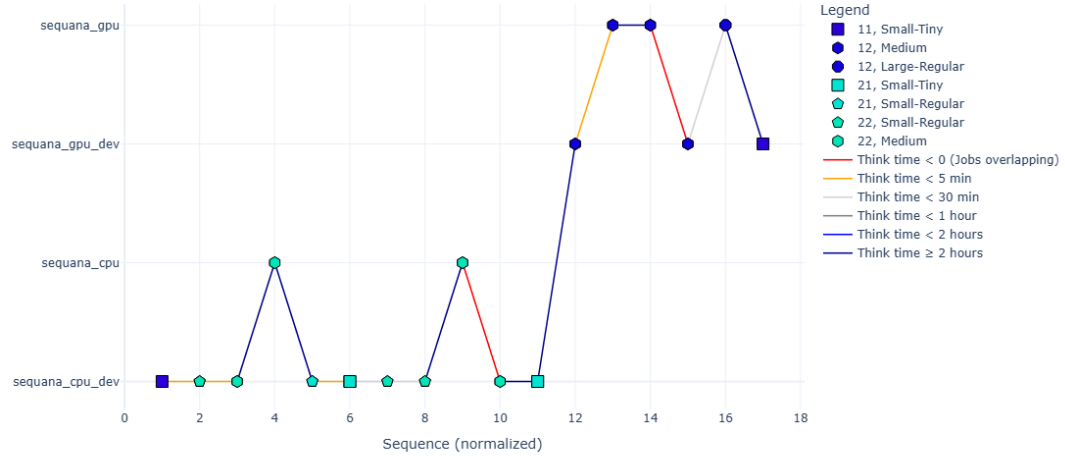


Figure 2. Individual trajectory over the first six months of 2023.

Figure 3 shows the graph $\mathbb{G}_{sim}$ and the modules detected by OSLOM in the original MAG, each color representing a set of vertices of $\mathbb{G}_{sim}$ within the same module. Users without connections are omitted, allowing the analysis to focus on group patterns. The configuration selected during the grid search has $t = 0.025$, $r = 1\,000$, and $cp = 0.10$. Across six seed pairs, the clustering agreement (adjusted Omega index) was 0.761 ± 0.076, indicating reasonably consistent module structures. Modal *Geometry* classes are: *Small-Mini* in module 5; *Small-Tiny* in module 0; *Small-Regular* in modules 2 and 7; *Medium* in modules 3 and 4; *Large-Regular* in module 1; and *Large-Extra* in modules 6, 8, and 9. GPU queues are modal in modules 6 and 7.

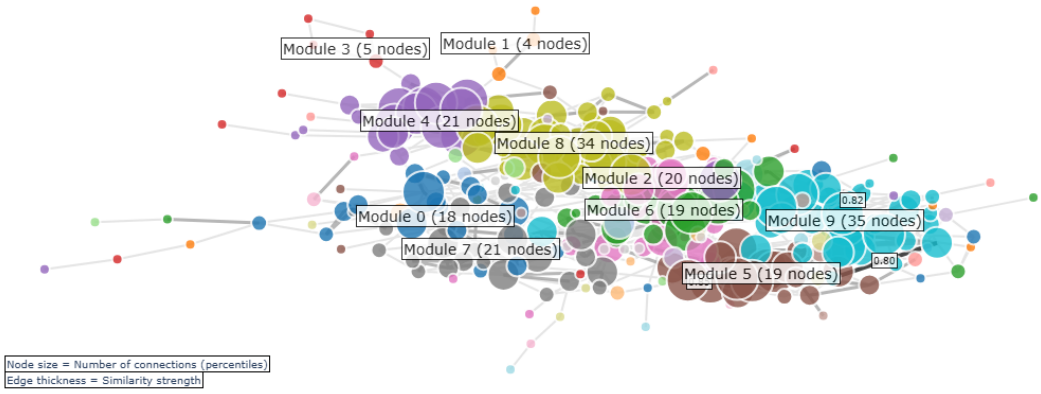


Figure 3. Modules detected using OSLOM over the first six months of 2023.

Figure 4 shows the trajectories of four users belonging to the same OSLOM module. In this figure, colors identify users, the Y-axis shows the aspect *Utilization*, and the "points" represent the aspect *Geometry*. This figure illustrates trajectories with recurring patterns of $Geometry \in \{\textit{Small-Mini}, \textit{Small-Tiny}\}$ and $Utilization = (1, 1)$.

5. Conclusion

In this work, we introduced a novel framework for modeling and categorizing user trajectories in supercomputing systems using Multi-Aspect Graphs (MAG). Our framework

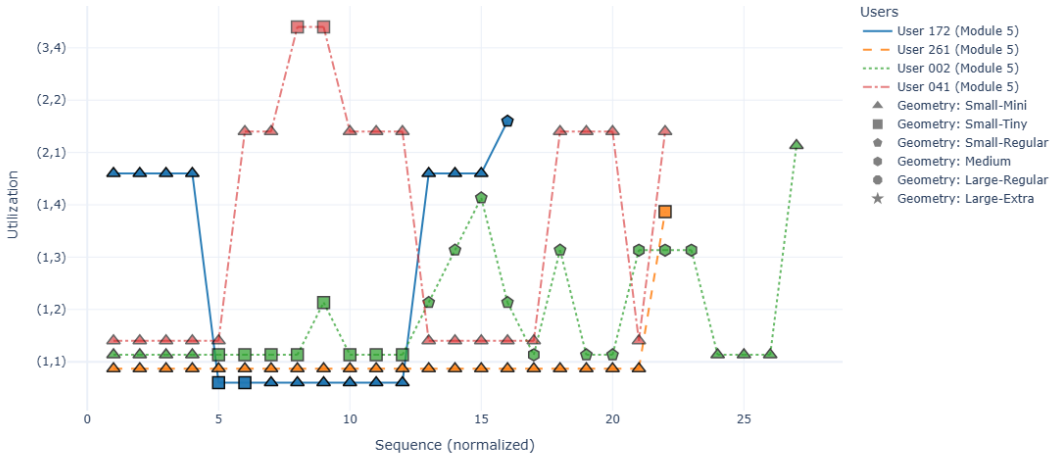


Figure 4. Multi-user trajectory (four in total) over the first six months of 2023.

integrates five complementary aspects of job submissions (resource requests, measured CPU and memory utilization, project context, submission chronology, and computational magnitude of jobs) into a unified representation with directed edges linking consecutive submissions and containing user identity and inter-submission think times. Moreover, the combination of a sequence alignment algorithm adapted for MAGs and graph-based overlapping cluster detection via OSLOM enables the analysis of individual trajectories and the identification of groups of users with similar behavioral profiles.

In our case study using six months of job logs from the Santos Dumont supercomputer, comprising 83 109 jobs submitted by 285 users, our framework preserved individual trajectories, captured concurrent submissions, revealed submission cadences, and identified recurring patterns of resource usage. The modules detected by OSLOM exhibited distinct characteristics in terms of job size and queue selection, with some modules showing prominent use of GPU queues. These findings demonstrate the potential of the framework to provide a more comprehensive representation of user behavior than isolated workload statistics, offering system administrators more context for guiding users and assessing scheduling policies.

Several directions remain for future work. First, to evaluate the robustness of our framework we must conduct a sensitivity analysis of its many parameters (aspect weights, alignment parameters, similarity-graph thresholds) when we consider other job logs. Second, we intend to incorporate detailed GPU utilization metrics (to account for more heterogeneous workloads) and filesystem metadata and I/O access patterns (for a broader view of data-intensive workflows) into the aspect design. Third, we intend to assess whether the framework provides measurable benefits over simpler workload-characterization approaches. Finally, we plan to integrate the framework with scheduling policy optimization [dos Santos et al. 2026], using observed profiles to generate realistic synthetic job logs that may improve the generality of optimized policies.

Use of Generative Artificial Intelligence

We used GPT-5.6 Sol (OpenAI) as a tool to help refining the text of this paper and examining some aspects of the research design presented in the introduction. We reviewed and evaluated all suggestions and are solely responsible for the content, methodology, analysis, interpretations, and conclusions presented in this paper.

References

Bose, A., Yang, H., Hsu, W. H., and Andresen, D. (2021). HPCGCN: A predictive framework on high performance computing cluster log data using graph convolutional networks. In *2021 IEEE International Conference on Big Data*, pages 4113–4118. IEEE.

Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27.

Collins, L. M. and Dent, C. W. (1988). Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions. *Multivariate Behavioral Research*, 23(2):231–242.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227.

dos Santos, J. P. M. N., da Silva, J. E. H., and Gomes, A. T. A. (2026). Towards workload-tailored optimization of job scheduling policies in HPC environments. *Concurrency and Computation: Practice and Experience*, 38(1):e70532.

dos Santos, J. P. M. N. and Gomes, A. T. A. (2024). Impact of job scheduling policy changes on user behaviour and system response: The case of the Santos Dumont supercomputer in Brazil. In *Latin American High Performance Computing Conference*. Springer.

Feitelson, D. G. (2001). Metrics for parallel job scheduling and their convergence. In Feitelson, D. G. and Rudolph, L., editors, *Job Scheduling Strategies for Parallel Processing*, pages 188–205, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jakobsche, T., Lachiche, N., and Ciorba, F. M. (2023). Investigating HPC job resource requests and job efficiency reporting. In *2023 22nd International Symposium on Parallel and Distributed Computing*, pages 61–68. IEEE.

Lancichinetti, A., Radicchi, F., Ramasco, J. J., and Fortunato, S. (2011). Finding statistically significant communities in networks. *PLOS ONE*, 6(4):e18961.

Lim, S.-H., Sim, H., Gunasekaran, R., and Vazhkudai, S. S. (2017). Scientific user behavior and data-sharing trends in a petascale file system. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, CA. University of California Press.

Madon, M., Costa, G. D., and Pierson, J.-M. (2024). Replay with feedback: How does the performance of HPC system impact user submission behavior? *Future Generation Computer Systems*, 155:66–79.

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.

Patel, T., Rich, P., Allcock, W., Liu, Z., Kettimuthu, R., and Tiwari, D. (2020). Job characteristics on large-scale systems: Long-term analysis, quantification, and implications.

In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE.

Rosa, C. O. C. S., Ito, M., Vieira, A. B., Wehmuth, K., and Gomes, A. T. A. (2023). Framework based on complex networks to model and mine patient pathways. *arXiv preprint arXiv:2309.14208*. arXiv:2309.14208v2 [cs.CY].

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

SchedMD (2023). *Slurm Workload Manager Documentation*. SchedMD LLC.

Schlagkamp, S., Ferreira da Silva, R., Allcock, W., Deelman, E., and Schwiegelshohn, U. (2016). Consecutive job submission behavior at Mira supercomputer. In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '16, page 93–96, New York, NY, USA. Association for Computing Machinery.

Simakov, N. A., White, J. P., DeLeon, R. L., Gallo, S. M., Jones, M. D., Palmer, J. T., Plessinger, B. D., and Furlani, T. R. (2018). A workload analysis of NSF's innovative HPC resources using XDMoD. *CoRR*, abs/1801.04306.

Wang, Q., Shen, Y., and Li, J. (2021). User-level workload analysis for supercomputers. In *Proceedings of the 4th International Conference on Software Engineering and Information Management*, pages 68–73. ACM.

Ward, Jr., J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

Wehmuth, K., Fleury, É., and Ziviani, A. (2016). On MultiAspect graphs. *Theor. Comput. Sci.*, 651:50–61.

You, H. and Zhang, H. (2013). Comprehensive workload analysis and modeling of a petascale supercomputer. In *Job Scheduling Strategies for Parallel Processing*, volume 7698 of *Lecture Notes in Computer Science*, pages 253–271. Springer.

Zheng, S., Shae, Z.-Y., Zhang, X., Jamjoom, H., and Fong, L. (2011). Analysis and modeling of social influence in high performance computing workloads. In *Euro-Par 2011 Parallel Processing*, volume 6852 of *Lecture Notes in Computer Science*, pages 193–204. Springer.