A Spatiotemporal Data Aggregation Technique for Performance Analysis of Large-scale Execution Traces

Damien Dosimont, Robin Lamarche-Perrin, Lucas Mello Schnorr, Guillaume Huard, Jean-Marc Vincent

A Spatiotemporal Data Aggregation Technique for Performance Analysis of Large-scale Execution Traces

Damien Dosimont*†‡§, Robin Lamarche-Perrin¶, Lucas Mello Schnorr‖, Guillaume Huard†‡*§ and Jean-Marc Vincent†‡*§
*Inria
†Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
‡CNRS, LIG, F-38000 Grenoble, France
§Email: firstname.lastname@imag.fr
¶MPI for Mathematics in the Sciences, 04103 Leipzig, Germany – Email: robin.lamarche-perrin@mis.mpg.de
‖Informatics Institute, UFRGS, Porto Alegre – Email: schnorr@inf.ufrgs.br

***Abstract*—Analysts commonly use execution traces collected at runtime to understand the behavior of an application running on distributed and parallel systems. These traces are inspected *post mortem* using various visualization techniques that, however, do not scale properly for a large number of events. This issue, mainly due to human perception limitations, is also the result of bounded screen resolutions preventing the proper drawing of many graphical objects. This paper proposes a new visualization technique overcoming such limitations by providing a concise overview of the trace behavior as the result of a spatiotemporal data aggregation process. The experimental results show that this approach can help the quick and accurate detection of anomalies in traces containing up to two hundred million events.**

***Keywords*-Performance analysis, trace visualization, spatio-temporal aggregation, information theory, NASPB, Grid'5000.**

I. Introduction

Large scale distributed systems represent typical cases of platforms involving several thousands computing resources. Even for a short computation time, the trace generated during an application execution on such systems may contain millions of events and reach several gigabytes of data. The events collected during the runtime can be function calls, interruptions, CPU load, memory utilization or hardware counters. Interpreted together, these events can help the identification of execution anomalies such as bad performance caused by resources or application bottlenecks.

Traces are commonly visualized by Gantt charts [1], which depict spatial and temporal dimensions and are capable to relate temporal behavior to hardware and software components. However, in large scale scenarios with many resources and events, users cannot be provided with a correct overview of the whole trace because of screen limitations. Figure 2 depicts all the events of a large trace, resulting in cluttered visualizations with very small graphical objects and pixelization artifacts [2]. By zooming in and panning to get more details, the analyst usually loses context, making it difficult to figure out which part of the trace is drawn and how representative it is considering the overall application behavior. However, if the user could dispose of a very large screen, data would be so numerous that it would be difficult to understand them. The limitation of a microscopic Gantt chart is that it is incapable to depict a *higher-level understanding* of the system states and dynamics. Because anomalies can be global (like bottlenecks), this high-level understanding is necessary to debug the system.

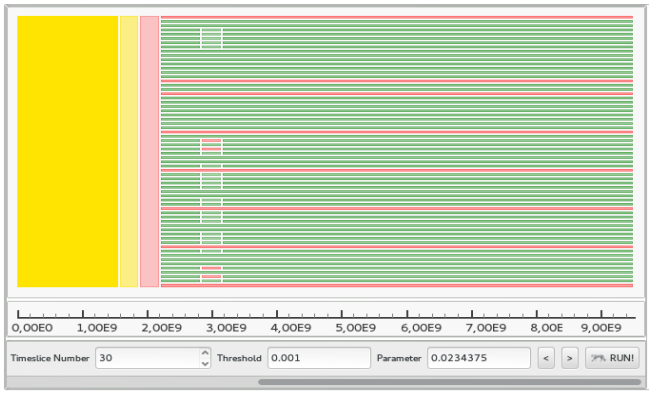


Figure 1. Our analysis tool, Ocelotl, showing an overview of the execution of the NAS-CG application, class C, 64 processes, on the Grid'5000 Rennes site: the trace is partitioned into aggregates that correspond to a locally homogeneous behavior of the application over time and among a set of computing resources. We distinguish a perturbation around 3,00E9, caused by the concurrent execution of applications competing for network access.

We propose an innovative visualization technique that provides a consistent overview of the temporal and resource dimensions. The main contribution consists in a multidimensional data-aggregation algorithm detecting and merging areas of the trace that are temporally and spatially homogeneous to provide higher-level visualizations while preserving the microscopic information content. In practice, this algorithm computes a partition of space and time that optimizes a trade-off between the representation complexity and information loss. The analyst can easily choose several levels of details by sliding the aggregation strength among a set of significant values. In addition to this algorithm, we present an associated visualization that provides a represen-

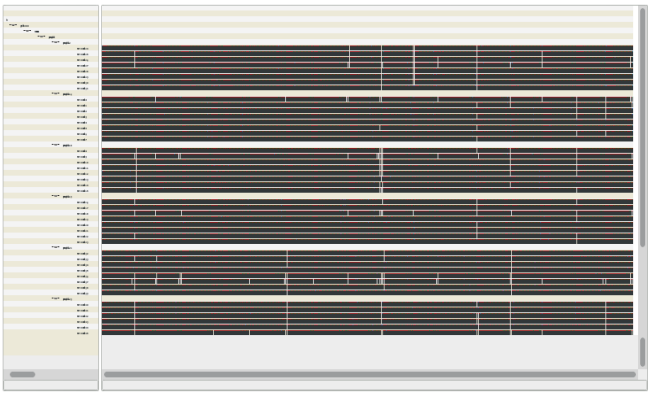


Figure 2. Example of a Gantt chart trying to represent the trace visualized in Figure 1. Even for a temporal subset of this trace ($1/7$), the visualization is cluttered because of the large amount of objects and rendering artifacts.

tative information of the aggregates by showing the mode (the value that appears most often), and a visual aggregation that completes the data aggregation procedure. Figure 1 presents a screenshot of our tool.

This paper is organized as follows. Section II shows related work on spatiotemporal visualizations tackling scalability. Section III describes our theoretical approach and the aggregation algorithm. Section IV gives the description of the visualization techniques for the algorithm output. Section V presents large scale MPI application experiments that validate our aggregation algorithm and corresponding visualization. Section VI concludes and details perspectives.

II. Related Work

Different analysis tools propose several methods to improve scalability and provide decent overviews over time and space. Table I gives a summary of these tools and the corresponding techniques.

A. Some Improper Complexity Reduction Techniques

Elmqvist and Fekete [3] propose a methodology to build an overview based on hierarchical aggregation. They insist on six criteria that a consistent visualization must fulfill:

- G1. *Entity budget*. The visual entity number is limited and entity size is big enough to avoid visual clutter.
- G2. *Visual summary*. Visual aggregates convey information about the underlying data.
- G3. *Visual simplicity*. Aggregates are clean and simple.
- G4. *Discriminability*. Aggregates are distinguishable from data items.
- G5. *Fidelity*. Aggregates does not lie about the size of an effect.
- G6. *Interpretability*. The aggregation process is interpretable by the user

On the following, we mark Gx if criterion x is satisfied, and $\overline{\text{Gx}}$ if it is not. Pixel-guided representations, present in some Gantt charts [4], [5], [6] or timelines [4], associate each pixel allocated for the view to a set of data. As the pixel is incapable to represent all the information it contains, the rendering algorithm decides which information is shown or hidden. We claim that this aggregation process is unclear for the user ($\overline{\text{G4}}$, $\overline{\text{G6}}$) and may mislead him about the trace content ($\overline{\text{G5}}$). We also notice that resizing the window modifies strongly the visualization content because pixel allocation changes, being unable to keep coherence between the representations. In what we call visual aggregation, the rendering tries as much as possible to preserve graphical object scales, whose size depends on their contents. When it is impossible, for instance, when the size is less than one pixel, it generates aggregates gathering close objects. In Pajé [7] and LTTng Eclipse Viewer [8], such aggregates are just used to avoid visual clutter, but do not represent the data they contain, which hinders the analysis ($\overline{\text{G2}}$).

B. Lack of Satisfying Spatiotemporal Overview

Even if some of them respect all the G criteria, the techniques presented in the Table I do not provide a fair processing of the temporal and spatial (resources) dimensions. This problem hinders the representation of phenomena that involve the dynamics of the application and the structure of the platform. We define two criteria, included in the Table I, to evaluate the management of spatiotemporal dimensions:

- M1. *Spatiotemporal representation*. The representation explicitly shows both spatial and temporal dimensions.
- M2. *Aggregation coherence*. The reduction process is applied simultaneously to both dimensions.

All representations based on Gantt charts respect M1. However, some of them neglect reduction techniques on the space [4], [5], [6], or use a technique depending on the dimension [9]. These lacks are strongly related to the incapability to fulfill the G1 criteria on both dimensions.

On the contrary, other techniques fulfill M2 but not M1. Vampir's task profile [4] clusters the most similar processes according to a distance measure based on the duration of the functions executed by each process. Even if the function distribution for each cluster is represented by a bar chart, the temporal dimension is lost in the process. Our previous work includes a way to manage the representation complexity and the information lost induced by an aggregation through an information-based compromise. Viva [13] provides a multiresolution treemap view showing the hardware and software component hierarchy. Entities having the most homogeneous behavior are aggregated using the information-based compromise. This technique helps to highlight troubles characterized by an heterogeneous behavior. Entities values (e.g., the function duration) are time-integrated over a configurable time interval. Nevertheless, the time dimension is missing from the representation ($\overline{\text{M1}}$). Ocelotl [11], [12] uses the same compromise to build a timeline where homogeneous time periods of the trace are aggregated, succeeding to highlight macroscopic phases and temporal perturbations.

Table I
SPATIO-TEMPORAL SCALABILITY TECHNIQUES IMPLEMENTED IN TRACE ANALYSIS TOOLS. THE GX ARE ELMQVIST AND FEKETE CRITERIA [3], MX ARE OUR SPATIOTEMPORAL CRITERIA. A CRITERION MIGHT BE SATISFIED: ONLY FOR TIME (⋆), SPACE (○), OR FOR BOTH DIMENSIONS (●)

Visualization	Technique	Tools	G1	G2	G3	G4	G5	G6	M1	M2
Gantt Chart	Pixel-guided (⋆), No aggregation (○)	Vampir [4], Paraver [5] [6]	⋆	●	●				●	
Gantt Chart	Visual Aggregation (⋆), No aggregation (○)	Pajé [7], LTTng Eclipse Viewer [8]	⋆		●	●	●	●	●	
Gantt Chart	Time compression (⋆), Hierarchical aggregation (○)	KPTrace Viewer [9]	○		●			●	●	
Gantt Chart	Time abstraction (⋆), No aggregation (○)	Jumpshot [10]	⋆	●	●	●	●	●	●	
Timeline	Pixel-guided (⋆, ○)	Vampir [4]	●	⋆	●					●
Timeline	Information aggregation (⋆, ○)	Ocelotl [11], [12]	●	●	●	●	●	●		●
Task Profile	Clustering (○), Mean Operation (⋆)	Vampir [4]	●	●	●	●	●	●		●
Treemap/Topology	Hierarchical aggregation (○), Time integration (⋆)	Viva [2], [13]	●	●	●	●	●	●		●

Even if spatial data is not represented ($\overline{\text{M1}}$), it is used to determine the time aggregates (M2).

C. Our contribution

Both Viva [13] and Ocelotl [11], [12] tools are highly compliant with the G criteria thanks to the information they provide to the user during the aggregation process, but not with the M1 criteria, since they represent only one dimension. This is why we propose to extend the technique to provide a satisfying overview (G) managing correctly and simultaneously the temporal and spatial dimensions (M). Two different processes are involved to address these requirements. A spatiotemporal *data aggregation* algorithm we present in Section III that reduces the trace complexity in both dimensions while controlling the information loss (G1, G5, G6, M1, M2). Section IV explains how its output is represented thanks to a visualization technique we have designed (G2, G3, G4), and how a *visual aggregation* improves the spatial entity budget management (G1).

III. A SPATIOTEMPORAL AGGREGATION ALGORITHM

A. The Trace Microscopic Model

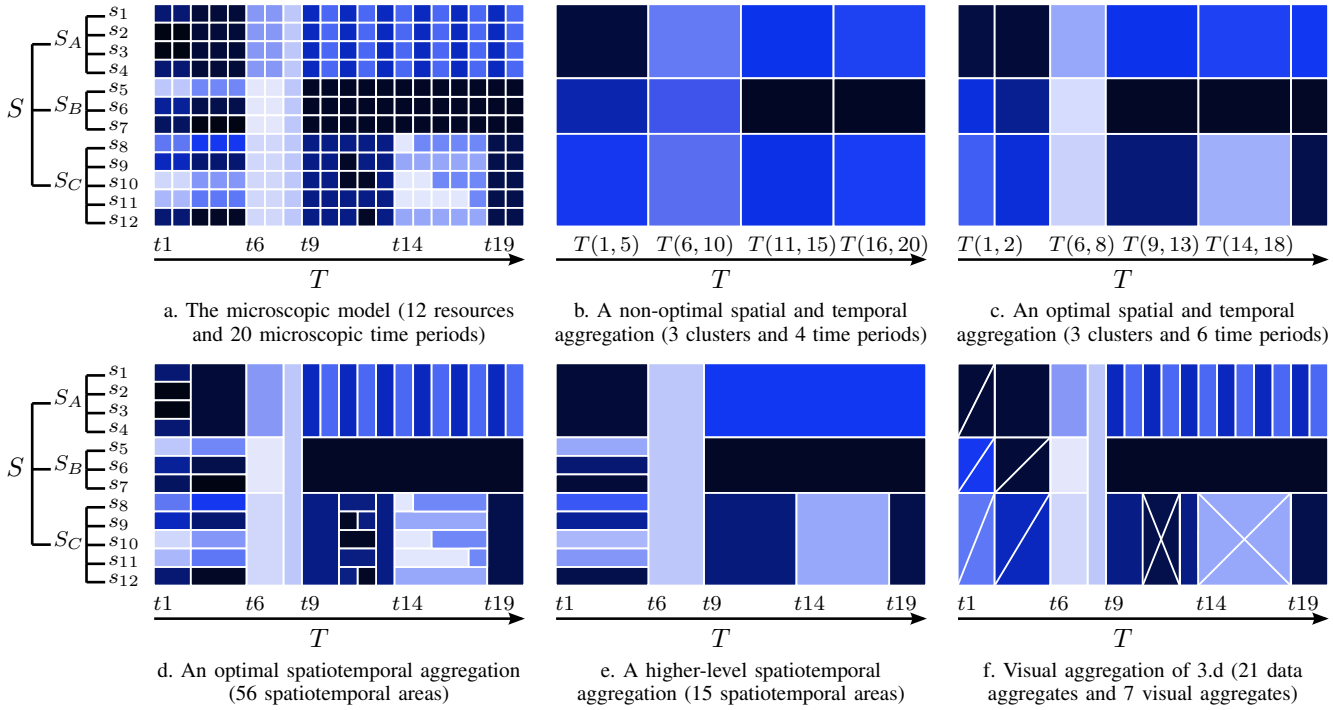
Raw traces contain timestamped events describing a particular behavior of the application, *e.g.*, a function call or its return, a communication, a synchronization (semaphore, mutex). Each event is associated with the resource that produces it, *e.g.*, a thread or a process. In order to provide an algorithmic framework for data aggregation, one has to properly formalize the trace dimensions, individuals and structures that will be actually aggregated. In our case, as we focus on the spatial and temporal dimensions, this model preliminary aggregates the events within microscopic spatiotemporal areas constituting what we call the trace *microscopic model*. It is thus described as algebraically-structured tridimensional datasets: the *spatial dimension* is structured according to the platform hierarchy, the *temporal dimension* is structured by the order of time, and the *state dimension* which, in this paper, has no particular structure.

(1) Spatial Dimension: The set $S = \{s_1, \ldots, s_n\}$ of the platform microscopic resources defines the trace *spatial dimension*. For the computing platforms we are interested in, this set has a hierarchical structure: resources are organized in processes, running on cores, each one being associated with a machine, themselves organized in clusters, and so on. Formally, a *hierarchy* is a set $\mathcal{H}(S) = \{S_1, \ldots, S_p\}$ of subsets of S that contains the whole resource set ($S \in \mathcal{H}(S)$), each singleton ($\forall s \in S,\ \{s\} \in \mathcal{H}(S)$), and such that any two parts in $\mathcal{H}(S)$ are either disjoint or included one in another ($\forall (S_i, S_j) \in \mathcal{H}(S)^2$, either $S_i \cap S_j = \emptyset$ or $S_i \subset S_j$ or $S_i \supset S_j$). A hierarchy is thus equivalent to a *rooted tree* where the *leaves* corresponds to the singletons, the *root* to the whole set, and the *tree-order* to the subset relation.

(2) Temporal Dimension: The set $T = \{t_1, \ldots, t_m\}$ of the observed microscopic time periods defines the *temporal dimension*. This dimension is discrete, whereas the raw trace time is continuous. To fit with the microscopic model, the raw trace is divided in $|T|$ (regular) time periods and the events are associated with the periods where they are active. Each period t has a duration $d(t) \in \mathbb{R}^+$ and the whole set is naturally ordered by "the arrow of time". Formally, a total order $<$ on T provides the concept of interval: $T_{(i,j)} = \{t \in T \mid t_i \leq t \leq t_j\}$ with $t_i \leq t_j$. We mark $\mathcal{I}(T)$ the set of intervals of T.

Spatiotemporal Dimension: The trace *spatiotemporal dimension* is given by the Cartesian product of the spatial and the temporal dimensions. A *microscopic* spatiotemporal area is thus a couple $(s, t) \in S \times T$ and a *macroscopic* spatiotemporal area is a subset of $S \times T$ that results from the Cartesian product of a node $S_k \in \mathcal{H}(S)$ and an interval $T_{(i,j)} \in \mathcal{I}(T)$. We mark $\mathcal{A}(S \times T) = \mathcal{H}(S) \times \mathcal{I}(T)$ the set of such areas with the following convention: "$S_k \times T_{(i,j)} \in \mathcal{A}(S \times T)$ is equivalent to $(S_k, T_{(i,j)}) \in \mathcal{H}(S) \times \mathcal{I}(T)$." The spatiotemporal dimension inherits both the hierarchical structure of space and the order of time: there is a hierarchy on $\{\{s\} \times T_{(i,j)}\}_{s \in S}$ for any interval $T_{(i,j)}$ and a total order on $\{S_k \times \{t\}\}_{t \in T}$ for any node S_k.

(3) State Dimension: The set $X = \{x_1, \ldots, x_l\}$ of the possible resource states defines the trace *state dimension*. A state is a timestamped event that has a start and an end. For instance, a function call and its termination forms a state. If this same function is called several times, we consider that it is the same state appearing multiple times. This metric gives information on the duration passed in the different actions performed by the resources, which is a way to define their behaviors quantitatively. In this paper, we renounce any particular algebraic structure on this set. Given a resource



a. The microscopic model (12 resources and 20 microscopic time periods)

b. A non-optimal spatial and temporal aggregation (3 clusters and 4 time periods)

c. An optimal spatial and temporal aggregation (3 clusters and 6 time periods)

d. An optimal spatiotemporal aggregation (56 spatiotemporal areas)

e. A higher-level spatiotemporal aggregation (15 spatiotemporal areas)

f. Visual aggregation of 3.d (21 data aggregates and 7 visual aggregates)

Figure 3. Aggregation and visualization of an artificial trace giving the behavior of 12 resources during 20 microscopic time periods (two possible states)

and a time period $(s,t) \in S \times T$, for each state $x \in X$, we mark $d_x(s,t) \in \mathbb{R}^+$ the total time spent in state x by s during t. We also mark $\rho_x(s,t) = d_x(s,t)/d(t) \in [0,1]$ the proportion of time spent in state x relatively to the period duration. Fig. 3.a presents such a trace with $|S| =$ 12 resources, $|T| = 20$ microscopic time periods, $|S \times T| =$ 240 spatiotemporal areas, and $|X| = 2$ possible states with proportions ρ_1 and ρ_2. For any resource s and time period t, the intensity of the corresponding square gives $\rho_1(s,t) = 1 - \rho_2(s,t)$.

B. The Trace Aggregation Model

Aggregation is a two-step abstraction process using a partition of the dataset to reduce the data it contains [3], [14], [15]. The first step thus consists in *partitioning* the entity set into disjoint and covering *aggregates*. Formally, a partition $\mathcal{P}(S)$ of S is a set of subsets that are pairwise disjoint ($\forall (S_i, S_j) \in \mathcal{P}(S)^2, S_i = S_j$ or $S_i \cap S_j = \emptyset$) and which union is the whole set ($\cup_{S_k \in \mathcal{P}(S)} S_k = S$). However, in order to be properly interpreted by the analyst, the aggregates should be consistent with the dataset algebraic structures [14]: resource aggregates are nodes of the platform hierarchy ($\mathcal{P}(S) \subset \mathcal{H}(S)$), time period aggregates are intervals of T ($\mathcal{P}(T) \subset \mathcal{I}(T)$), and spatiotemporal aggregates are each the Cartesian product of a node and an interval ($\mathcal{P}(S \times T) \subset \mathcal{H}(S) \times \mathcal{I}(T)$). We mark $\boldsymbol{\mathcal{H}}(S)$ the set of *hierarchy-consistent partitions*, $\boldsymbol{\mathcal{I}}(T)$ the set of *order-consistent partitions*, and $\boldsymbol{\mathcal{A}}(S \times T)$ the set of *hierarchy-and-order-consistent partitions*. Note that, in this section, we disregard the aggregation of the state dimension. The second step of the aggregation process consists in *reducing* the microscopic data and providing an overview according to the chosen partition. Micro-entities are then replaced by virtual macro-entities and their properties are condensed by an *aggregation operator* (*e.g.*, sum, mean, extrema [3]). In the spatiotemporal case, the state proportions are simply averaged to get the overall state proportions of the aggregated areas: $\forall (S_k, T_{(i,j)}) \in \mathcal{H}(S) \times \mathcal{I}(T)$, $\forall x \in X$,

$$\rho_x(S_k, T_{(i,j)}) = \frac{1}{|S_k|} \sum_{s \in S_k} \left(\frac{\sum_{t \in T_{(i,j)}} d_x(s,t)}{\sum_{t \in T_{(i,j)}} d(t)} \right) \qquad (1)$$

where $|S_k|$ is the number of underlying resources.

Fig. 3.b gives an example of spatiotemporal aggregation of the microscopic trace represented in Fig. 3.a. The spatial dimension is aggregated according to the second level of the hierarchy ($\mathcal{P}(S) = \{S_A, S_B, S_C\}$) and the temporal dimension according to four intervals of five microscopic time periods ($\mathcal{P}(T) = \{T_{(1,5)}, T_{(6,10)}, T_{(11,15)}, T_{(16,20)}\}$). In this example, the spatiotemporal partition is simply the Cartesian product of these unidimensional partitions: $\mathcal{P}(S \times T) = \mathcal{P}(S) \times \mathcal{P}(T)$.

C. Quantifying Data Reduction and Information Loss

As briefly addressed in previous sections and in further details in previous work [14], [16], aggregation is a complex process and it may turn to be harmful for the analysis if the partition is poorly chosen. In particular, the reduction step may cause misleading information losses. For example, in Fig. 3.b, aggregate $(S_1, T_{(1,5)})$ seems quite adequate to summarize the corresponding microscopic data since

the state proportions are homogeneous in space and time ($\forall (s,t) \in (S_1, T_{(1,5)}),\ \forall x \in X,\ \rho_x(s,t) \approx \rho_x(S_1, T_{(1,5)})$). On the contrary, $(S_3, T_{(6,10)})$ is an inappropriate approximation since some significant variations exist between the aggregated state proportions and the real microscopic ones.

In order to provide an overview that can be properly interpreted by the analyst, one has to aggregate as a priority the homogeneous spatiotemporal areas, while preserving the microscopic information regarding the heterogeneous ones. Henceforth, aggregation consists in optimizing a trade-off between data reduction and information loss. Information-theoretic measures have been proposed in previous work [16], [13], [11], [12] to express such a trade-off: Kullback-Leibler divergence [17] as a measure of information loss:

$$\mathrm{loss}_x(S_k, T_{(i,j)}) = \sum_{(s,t)\in(S_k,T_{(i,j)})} \rho_x(s,t) \log_2 \left(\frac{\rho_x(s,t)}{\rho_x(S_k, T_{(i,j)})} \right) \quad (2)$$

Shannon entropy [18] as a measure of data reduction:

$$\begin{aligned} \mathrm{gain}_x(S_k, T_{(i,j)}) = \rho_x(S_k, T_{(i,j)}) \log_2 \rho_x(S_k, T_{(i,j)}) \\ - \sum_{(s,t)\in(S_k,T_{(i,j)})} \rho_x(s,t) \log_2 \rho_x(s,t) \end{aligned} \quad (3)$$

and a *parametrized Information Criterion* [16]:

$$\mathrm{pIC}_x = p\,\mathrm{gain}_x - (1-p)\,\mathrm{loss}_x \quad (4)$$

where $p \in [0,1]$ is the gain/loss ratio used to balance this trade-off. For $p = 0$, the analyst wants to be as accurate as possible (the microscopic partition is optimal) and, for $p = 1$, she wants to be the simplest (the full aggregation is optimal). When p varies from 0 to 1, a whole class of nested representations arises. The choice of this parameter is deliberately left to the analyst, so she can adapt the entity budget to the analysis purposes. Note that this criterion is additive according to the partitioning process [16] and to the state dimension [11], [12]:

$$\mathrm{pIC}(\mathcal{P}) \;=\; \sum_{P\in\mathcal{P}} \mathrm{pIC}(P) \;=\; \sum_{P\in\mathcal{P}} \sum_{s\in X} \mathrm{pIC}_x(P)$$

D. Spatial and Temporal Aggregation

Aggregation is thus modeled as a constrained optimization problem: *which structure-consistent partition maximizes the parametrized information criterion?* The computation of optimal hierarchy-consistent partitions have been addressed for multilevel detection of communities in large networks [19], for spatial analysis of geographic information systems [16], and for anomaly detection in the execution trace of distributed systems [13]. The computation of optimal order-consistent partitions have been addressed for temporal aggregation of time series [20], for temporal analysis of geographic information systems [16], and for anomaly detection in the execution trace of multimedia [11], [12] or MPI applications [12]. It has been shown that these algorithms belong to a larger class of optimization algorithms which can be formalized within a unified conceptual framework [14], [15]. The spatiotemporal aggregation algorithm we propose in the next subsection consists in a combination of the spatial and temporal algorithms and thus belongs to the same class.

Algebraic Structure of the Partition Sets: The numbers of hierarchy-consistent and order-consistent partitions both grow exponentially with the size of the corresponding dimension [15]: $|\boldsymbol{\mathcal{H}}(S)| = \Theta(c^{|S|})$, where $c \approx 1.229$ corresponds to the worst case scenario (the hierarchy is a complete binary tree), and $|\boldsymbol{\mathcal{I}}(T)| = O(2^{|T|})$. Hence, a brute-force search is intractable in practice.

In order to provide a computationally-efficient optimization algorithm, one first needs to acknowledge the algebraic structure of the solution spaces $\boldsymbol{\mathcal{H}}(S)$ and $\boldsymbol{\mathcal{I}}(T)$. They are partially ordered by the *refinement relation* [15]. This partial order allows to cleverly decompose the search space and to design multi-branching recursive algorithms that efficiently solve specialized versions of the optimization problem [14], [15]. In this context, the partitioning of hierarchical systems aims at finding the optimal level on each branch of the hierarchy by performing a simple depth-first search of the corresponding tree [19], [13], [16]. An optimal hierarchy-consistent partition is thus computed in linear time $O(|S|)$. The partitioning of ordered systems aims at cutting the time series by sequentially computing optimal subpartitions of growing subsets of the overall population (dynamic programming) [20], [11], [12], [16]. An optimal order-consistent partition is thus computed in quadratic time $O(|T|^2)$.

Spacial-and-temporal is not Spatiotemporal: Fig. 3.c gives an example of partition combining the independent results of the two unidimensional algorithms. Spatial aggregation is applied to the temporally-aggregated trace $S \times \{T\}$ and results in a hierarchy-consistent partition $\mathcal{P}(S)$ that aggregates the nodes where the underlying resources have spatially-homogeneous state proportions. Temporal aggregation is applied to the spatially-aggregated trace $\{S\} \times T$ and results in a order-consistent partition $\mathcal{P}(T)$ that divides the temporal dimension into periods where the state proportions are temporally-homogeneous. Then, the partition $\mathcal{P}(S \times T) = \mathcal{P}(S) \times \mathcal{P}(T)$ is used to aggregate the set of microscopic spatiotemporal areas $S \times T$.

Although the resulting aggregated representation is better in terms of information content than the one presented in Fig. 3.b (in particular, the temporal partition is much more tuned to the microscopic data), this technique suffers from two severe limitations. Firstly, the spatial algorithm computes an optimal partition of S without taking into account any temporal information (and conversely for the temporal algorithm). Hence, the state proportions may be averaged in such a way that the information content of the two unidimensional datasets $S \times \{T\}$ and $\{S\} \times T$ is lower than the one of the spatiotemporal dataset $S \times T$ (for further investigation, the mutual information would be an adequate measure to quantify this information loss). Secondly, and

more generally, some spatiotemporal patterns cannot be expressed as the Cartesian product of two unidimensional partitions: $\boldsymbol{\mathcal{H}}(S) \times \boldsymbol{\mathcal{I}}(T) \subset \boldsymbol{\mathcal{A}}(S \times T)$. Fig. 3.d gives examples of such patterns: $T_{(1,2)}$ is *homogeneous* in time and *heterogeneous* in space; $T_{(3,5)}$ is *homogeneous* in time and *heterogeneous* in space except for cluster S_A; $T_{(6,7)}$ is *homogeneous* in time and in space (at the cluster level); $T_{(8)}$ is fully *homogeneous* in time and in space; and within $T_{(9,20)}$: S_A is *homogeneous* in space and *heterogeneous* in time; S_B is *homogeneous* in space and in time; S_C contains more complex imbrications of homogeneous and heterogeneous spatial and temporal patterns. In order to overcome this limitation, the next subsection proposes a spatiotemporal algorithm that directly compute an optimal partition of $\boldsymbol{\mathcal{A}}(S \times T)$.

E. Our Spatiotemporal Aggregation Algorithm

Data Structure: In the aforementioned spatial aggregation algorithm, the hierarchy $\mathcal{H}(S)$ is stored as a tree data structure such that each $node$ corresponds to a hierarchical part S_k. In the temporal aggregation algorithm, the set of intervals $\mathcal{I}(T)$ is stored as a upper triangular matrix such that each cell $[i, j]$ corresponds to an interval $T_{(i,j)}$ [15]. The spatiotemporal aggregation algorithm we propose mixes these two data structures such that the set of spatiotemporal areas $\mathcal{A}(S \times T) = \mathcal{H}(S) \times \mathcal{I}(T)$ is stored as tree of upper triangular matrices such that each cell $[i, j]$ of each $node$ corresponds to an area $(S_k, T_{(i,j)})$.

Data Input: We represent the data related to the spatiotemporal area $(S_k, T_{(i,j)})$ as follows:

- the information loss: $node.loss[i, j] = \text{loss}(S_k, T_{(i,j)})$;
- the data reduction: $node.gain[i, j] = \text{gain}(S_k, T_{(i,j)})$.

some intermediary data required to compute the loss and the gain (see Eq. 2 and 3):

- the aggregated proportions: $\forall x \in X,\ \rho_x(S_k, T_{(i,j)})$;
- the sum of the state proportions of the underlying areas: $\forall x \in X,\ \sum_{(s,t)\in(S_k,T_{(i,j)})} \rho_x(s, t)$;
- the sum of the "Shannon information" of these proportions: $\forall x \in X,\ \sum_{(s,t)\in(S_k,T_{(i,j)})} \rho_x(s, t) \log_2 \rho_x(s, t)$;

and some other intermediary data required to compute the aggregated state proportions (see Eq. 1):

- the number of underlying resources: $|S_k|$;
- the total duration of the time period: $\sum_{t\in T_{(i,j)}} d(t)$;
- the time spent in each state by the resources during this time period: $\forall x \in X,\ \sum_{(s,t)\in(S_k,T_{(i,j)})} d_x(s, t)$.

The computation of these input data is not detailed here since it consists in the direct application of Eq. 1, 2, and 3. Iterations over the cells of the matrices, from the first line to last one, and nested in a tree recursion, from the leaves to the root, allow to compute these data in time $O(|S||T|^2|)$.

Data Output: Any partition $\mathcal{P}(S \times T) \in \boldsymbol{\mathcal{A}}(S \times T)$ can be represented as a sequence of *nested cuts* of the spatiotemporal areas in $\mathcal{A}(S \times T)$. We distinguish *spatial cuts*, when the area $(S_k, T_{(i,j)})$ is partitioned according to the immediate lower level in the hierarchy $\{(S_{k_1}, T_{(i,j)}), \ldots, (S_{k_q}, T_{(i,j)})\}$, where $S_{k_1}, \ldots, S_{k_q}$ are the children of S_k, and *temporal cuts*, when the area is partitioned into two intervals $\{(S_k, T_{(i,cut)}), (S_k, T_{(cut+1,j)})\}$, where $i \leq cut < j$. Note that a sequence of cuts uniquely defines a partition $\mathcal{P}(S \times T)$, whereas a given partition may be expressed according to different sequences. Hence, our spatiotemporal aggregation algorithm recursively computes, for each cell of each node of the data structure, a *cut value* corresponding to a step in the sequence of cuts defining an optimal partition of the corresponding area:

- $node.cut[i, j] = -1$ indicates a spatial cut;
- $node.cut[i, j] \in \{i, \ldots, j-1\}$ indicates a temporal cut (the integer then gives the index of the microscopic time period where the cut occurs);
- $node.cut[i, j] = j$ indicates that $(S_k, T_{(i,j)})$ is an aggregate of the partition ("no cut" case).

After the algorithm execution, the resulting optimal partition is thus recovered by looking at the sequence of cuts beginning from $(S_{root}, T_{(0,|T|-1)})$ and looking at the underlying areas until each aggregate of the partition have been reached. In order to compute these cut values, the algorithm also computes $node.pIC[i, j]$, which is the value of the information criterion of an optimal partition of $(S_k, T_{(i,j)})$.

Algorithm Description: The spatiotemporal algorithm computes the optimal cuts depending on the optimal partitions of the underlying areas. It consists in cell-iterations to find temporal cuts, nested in a depth-first search of the hierarchy to find spatial cuts. *Recursion:* For each spatiotemporal area $(S_k, T_{(i,j)})$, the pIC of the corresponding aggregate (see Eq. 4) is compared to the sum of the (previously-recursively-computed) pIC of the optimal partitions of the children of S_k (sum of $child.pIC[i, j]$). A spatial cut is detected if the former is lower than the latter. *Iteration:* For each possible $cut \in \{i, \ldots, j-1\}$, the pIC is then compared to the sum of the (previously-iteratively-computed) pIC of the optimal partitions of $(S_k, T_{(i,cut)})$ and $(S_k, T_{(cut+1,j)})$. A temporal cut is detected if one of these pIC is higher than the others. This way, all possible cuts are evaluated, and so are all possible partitions. Fig. 3.d and 3.e present two resulting optimal partitions computed by the algorithm for two different values of the gain/loss ratio parameter ($p_d < p_e$). Each spatiotemporal area of these partitions is then homogeneous relatively to the corresponding description level.

Algorithmic Complexity: Since each cell of each node contains a bounded number of data, the spatial complexity of the algorithm is $O(|S||T|^2)$. The overall time needed for the detection of spatial cut is linear with the number of nodes in the hierarchy (one addition and one comparison for each node). Given a node, for each cell $[i, j]$ of that node, with $0 \leq i \leq j < |T| - 1$, the algorithm performs $j - i$ detections of possible temporal cut (each in bounded time). Hence, a total of $\sum_{k=0}^{|T|-1} i\ (|T| - i) = O(|T|^3)$ detections

are performed for each node and the overall time complexity of the spatiotemporal aggregation algorithm is $O(|S||T|^3)$.

Algorithm 1 computes a hierarchy-and-order-consistent partition that maximizes the parametrized information criterion

```
procedure node.COMPUTEOPTIMALPARTITION(p)
    for each child do                                   ▷ Recursion
        child.COMPUTEOPTIMALPARTITION(p)

    for i = |T| − 1, ..., 0 do                          ▷ Iteration
        for j = i, ..., |T| − 1 do

            cut[i, j] ← j                               ▷ No cut
            pIC[i, j] ← p.gain[i, j] − (1 − p).loss[i, j]

            if has children then                        ▷ Spatial cut?
                pIC_s ← 0
                for each child do
                    pIC_s ← pIC_s + child.pIC[i, j]
                if pIC_s > pIC[i, j] then
                    cut[i, j] ← −1
                    pIC[i, j] ← pIC_s

            for cut_t = i, ..., j − 1 do                ▷ Temporal cut?
                pIC_t ← pIC[i, cut] + pIC[cut + 1, j]
                if pIC_t > pIC[i, j] then
                    cut[i, j] ← cut_t
                    pIC[i, j] ← pIC_t
```

IV. Visualizing the Algorithm Output

Figures 3.d and 3.e, which correspond to the algorithm output for different values of the gain/loss ratio p, meet criteria G2 (aggregates show homogeneous state proportions), G3 (rectangles are clear and simple), G4 (we easily distinguish aggregates from microscopic data thanks to their size), G5 (complexity reduction and information loss are provided to the user to indicate how far the representation is from the microscopic model), and G6 (aggregation is interpreted as a process to detect spatiotemporal areas with homogeneous behavior). M1 and M2 are also fulfilled (both dimensions are represented; the aggregation process handles them symmetrically and simultaneously).

However, these representations only deal with microscopic models that contains only two possible states ($|X| = 2$). When it contains more ($|X| > 2$), showing all the corresponding state proportion within a given area may introduce a visual clutter ($\overline{\text{G3}}$). In order to keep providing useful information (G2), we first associate a color col_x to each state $x \in X$. Then, each aggregate $(S_k, T_{(i,j)})$ represent the state mode, i.e., the state color that has the highest proportion: $x_{max} = \arg\max_{x \in X} \rho_x$. Last, we also provide information regarding the state mode proportion by applying transparency α during the rendering to change color intensity: $\alpha = \rho_{x_{max}} / \sum_{x \in X} \rho_x \in [1/|X|, 1]$. Figure 1 shows the result of this choice. Three state modes `MPI_init` (yellow), `MPI_send` (green) and `MPI_wait` (red) are visible.

When the number of resources $|S|$ is greater than the amount of pixels in the spatial axis, the visual entity budget is not respected for the microscopic representation and may also be problematic for small spatial aggregates ($\overline{\text{G1}}$). We use visual aggregation (during rendering) to maintain this budget: if an aggregate has a visual height inferior to a threshold (in pixels), its parent is drawn instead. Figure 3.f shows the difference between *data* and *visual* aggregation. Visually-aggregated areas are marked differently (G4): by a diagonal line, if underlying resources have the same temporal data partitioning, or by a cross, on the contrary.

Table II
Scenarios description and Ocelotl execution times

	Case A	Case B	Case C	Case D
Application	CG, class C	CG, class C	LU, class C	LU, class B
Processes	64	512	700	900
Site	Rennes	Grenoble	Nancy	Rennes
Clusters (nodes)	parapide(8)	adonis(9), edel(24), genepi(31)	graphene(26), graphite(4), griffon(67)	parapide(21), parapluie(18)
Event number	3,838,144	49,149,440	218,457,456	177,376,729
Trace size	136.9 MB	1.8 GB	8.3 GB	6.7 GB
Ocelotl computation times				
Trace reading	44 s	613 s	2911 s	2091 s
Microscopic description	4 s	55 s	244 s	196 s
Aggregation	<1s	<1s	2s	2s

V. Experimental Validation with the NASPB

We validate Ocelotl with the CG and LU parallel algebraic applications of the NAS Benchmark [21]. We run them with various settings on different sites of Grid'5000[1] and we trace the MPI function calls with Score-P[2]. The resource hierarchy is as follows: cores (up to 900) are grouped by machines (8 to 97 machines), which are grouped by clusters (1 to 3 clusters), that are finally grouped by site. During the execution, each MPI process is bound to a core. Table II shows the execution settings of four scenarios, with data about the generated traces, and the computation times with Ocelotl (these times are obtained on a PC with 4 cores Intel Xeon CPU E3-1225 v3 at 3.20 GHz, 32 GB DDR3, 256 GB SSD). The microscopic model is each time composed by 30 timeslices. We present cases A and C where an undesired behavior of the application has been detected by the aggregation algorithm. Cases B and D are used only to provide information on analysis computation times.

A. Highlight a Temporal Perturbation on a Small Use Case

We start by a small use case of 64 cores to show the result without visual aggregation. Our objective is to determine the temporal behavior of an application when it is expected to be regular. We also wish to compare the process behavior over the space dimension. NASPB-CG is frequently used to evaluate machine performance [21]. It solves an unstructured sparse linear system by the conjugate gradient method. It tests irregular long distance communication and employs

[1]https://www.grid5000.fr/mediawiki/index.php/Special:G5KHardware
[2]http://www.vi-hps.org/projects/score-p

unstructured matrix multiplication. We run CG, class C, on 64 cores (8 per machine), on the Parapide cluster of Rennes site (Table II, case A). Since the nodes have Infiniband cards MT25418, we expect regular and quick communications.

Figure 1 shows the visualization provided by Ocelotl using the spatiotemporal aggregation algorithm for case A. We easily distinguish an initialization phase, composed by `MPI_init` function calls, represented by the algorithm as a unique spatiotemporal data aggregate (all clusters, from 0 s to 1.6 s). This phase is followed by two spatially-aggregated time periods (from 1.6 s to 1.9 s and from 1.9 s to 2.2 s), corresponding to a transition into the computation phase. In this first phase, the aggregation thus highlights that the resources have approximately the same behavior. On the contrary, the computation phase (2.2 s – 9.5 s) is represented by temporal aggregates globally showing a regular behavior through time for each resource, but are different among the resources. Each 8-core machine has a process dedicated to `MPI_wait` function calls while the others are mainly running `MPI_send`. We also detect a perturbation around 3s after the application start. It is visualized as disruptions in the temporal aggregation of 26 processes. By analyzing this time period with a traditional Gantt chart, we discover that some `MPI_send` and `MPI_wait` last longer than in the rest of the trace, contradicting our expectations. Even with exclusive access to the nodes, the other machines of the cluster might be allocated to other users during the experience. The network is therefore a point of contact among experiments, inducing local perturbations. By running several executions with different settings (site, clusters), this anomaly appears occasionally, and never at the same moment in the trace.

Henceforth, Ocelotl easily detects temporal perturbations that would be difficult to detect with traditional analysis techniques such as statistics, because the size of the effect is sufficiently small to not affect the total execution time. We are also able to acknowledge the fact that all processes are not equally impacted by the perturbation, and gives a detailed list of those who significantly are. This would not be possible with the approach of previous work [12], [16].

B. Scalability of the Analysis on a Multi-cluster Application

The goal here is to show the scalability of our technique on a 218 million-event and 700-process trace. We wish to confirm that our aggregation still gives relevant information on temporal and spatial dimensions despite the quantity of data contained in the trace. Since the hardware where the application is executed is heterogeneous, we expect to find this characteristic on the spatial behavior. LU, for Lower-Upper Gauss-Seidel solver, solves a synthetic system of nonlinear PDEs, employing an algorithm that involves a symmetric successive over-relaxation (SSOR) kernel [21]. We run LU, class C, on 700 cores heterogeneously distributed between three clusters of the Nancy site (Table II, case C). The hardware is heterogeneous. In particular, the Graphite cluster has 10 Gigabit Ethernet cards and 16 cores per machine, whereas the two other clusters have Infiniband-20G interconnects, and 4 cores (Graphene) or 8 cores (Griffon) per machine.

Figure 4 shows an initialization phase (`MPI_init`, 0 s-17.5 s) followed by a spatially-heterogeneous phase containing `MPI_Allreduce` function calls. The computation phase starts at almost 20s and shows a curious behavior. The three clusters are separated by the aggregation algorithm. The temporal and spatial behavior of the Graphene cluster is homogeneous over the whole computation phase. On the contrary, the Graphite nodes are all spatially separated. The diagonal line indicates that even when visually aggregated, the behavior of these processes is heterogeneous. Last, we see a strong rupture at 34.5s in Griffon, whereas the rest of the computation phase is spatio-temporal aggregated.

Several trails can explain these phenomena. We first suspect the slower Graphite network (Ethernet) to hinder the communications. By zooming in on the Gantt chart, we detect frequent long `MPI_wait` and `MPI_send` with irregular patterns. We rely this behavior to the network performance. Regarding the temporal perturbation of the Griffon cluster, we also detect that two machines are blocked twice in a `MPI_wait` and two others are simultaneously blocked in `MPI_send`. For this second case, the cluster internal network is accessed through Infiniband, and we do not expect bad communication performance. Moreover, Griffon was completely allocated to us during the program execution. By investigating, we have discovered that Griffon switches are shared with other machines that are not accessible for Grid'5000 normal users. However, these machines are active and they access the network. A concurrency phenomenon may explain this temporal perturbation.

Hence, Ocelotl enables to detect, within a very large trace (218 million of events), a spatial and temporal abnormal behavior that we were unable to show with other tools (as Gantt charts) for performance reasons. Ocelotl manages correctly the visualization of a high amount of resources (700 cores), and the aggregation still stays precise despite the high quantity of information contained in the trace.

VI. Conclusion and future work

We have presented a visualization technique, based on a spatiotemporal aggregation, which provides a macroscopic description of a trace and can be used as an entry point to the analysis. We fulfill criteria [3] that define a good quality overview, and keep representing spatial and temporal dimension by using an aggregation that involves them coherently. We present two use cases that highlight the efficiency of our technique to synthesize the trace behavior of realistic application executions. A small use case shows a temporal perturbation in a MPI application, typical of a concurrency situation to access the network, but difficult to find with statistical methods. A larger one, with 700 processes and 218

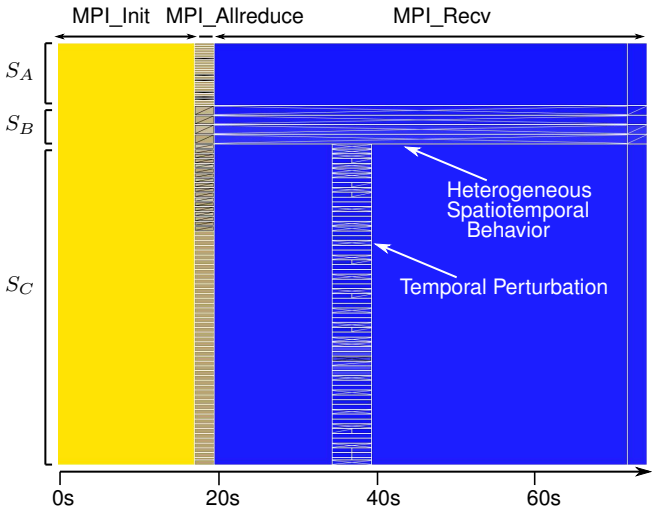


Figure 4. Ocelotl overview of the MPI application LU, class C, 700 processes, executed on the Nancy site of Grid'5000 (S_A: Graphene, S_B: Graphite, S_C: Griffon). We mainly distinguish an initialization sequence (0-20 s), followed by the computation phase, where the behavior of the Graphite cluster is heterogeneous in space and time, and there is a perturbation that touches only the execution of the Griffon cluster (34.5 s).

million events, demonstrates the scalability of our approach by highlighting the influence of the hardware characteristics on the process behavior. We also detect a temporal perturbation caused by machines that were hidden to the user. For this case, the tool we have designed enables to keep good performance: a 50 min preprocess is needed to load this large trace, but then, we provide an instantaneous interaction to get the visualization at a given aggregation level.

Our future work is focused on the improvement of our visualization to go further to fulfill the overview criteria: in particular, we are interested in conveying more information with the aggregates that compose our visualization, and we foresee to use interaction solutions to retrieve data such as the proportion of all the active states. We consider that our usage of the transparency could be improved to better differentiate states, since its effect on the user is dependent on the colors that are employed. Solutions using different color spaces, as YCbCr, could be employed.

References

[1] J. Wilson, "Gantt Charts: A Centenary Appreciation," *European Journal of Operational Research*, vol. 149, no. 2, 2003.

[2] L. M. Schnorr, A. Legrand, and J.-M. Vincent, "Detection and Analysis of Resource Usage Anomalies in Large Distributed Systems Through Multi-Scale Visualization," *Concurrency and Computation*, vol. 24, no. 15, 2012.

[3] N. Elmqvist and J.-D. Fekete, "Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, 2010.

[4] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, and W. E. Nagel, "The Vampir Performance Analysis Tool-Set," in *Tools for High Performance Computing*. Springer-Verlag Berlin, 2012.

[5] V. Pillet, J. Labarta, T. Cortes, and S. Girona, "Paraver: A tool to visualize and analyze parallel code," in *Proceedings of WoTUG: Transputer & Occam Developments*, vol. 44, 1995.

[6] J. Labarta, J. Gimenez, E. Martinez, P. González, H. Servat, G. Llort, and X. Aguilar, "Scalability of visualization and tracing tools," in *Proceedings of Parallel Computing*, 2005.

[7] J. Chassin de Kergommeaux, "Pajé, an Interactive Visualization Tool for Tuning Multi-Threaded Parallel Applications," *Parallel Computing*, vol. 26, no. 10, 2000.

[8] "Linux Tools Project/LTTng2/User Guide - Eclipsepedia," http://wiki.eclipse.org/index.php/Linux_Tools_Project/LTTng2/User_Guide.

[9] C. Prada-Rojas, F. Riss, X. Raynaud, S. De Paoli, and M. Santana, "Observation tools for debugging and performance analysis of embedded linux applications," in *Conference on System Software, SoC and Silicon Debug-S4D*, 2009.

[10] E. Lusk and A. Chan, "Early Experiments with the OpenMP/MPI Hybrid Programming Model," in *OpenMP in a New Era of Parallelism*, ser. LNCS, vol. 5004. Springer-Verlag Berlin, 2008.

[11] G. Pagano, D. Dosimont, G. Huard, V. Marangozova-Martin, and J.-M. Vincent, "Trace Management and Analysis for Embedded Systems," in *Proceedings of the IEEE 7th International Symposium on Embedded Multicore SoCs*, 2013.

[12] D. Dosimont, L. M. Schnorr, G. Huard, and J.-M. Vincent, "A Trace Macroscopic Description based on Time Aggregation," INRIA, Tech. Rep. RR-8524, 2014.

[13] R. Lamarche-Perrin, L. M. Schnorr, J.-M. Vincent, and Y. Demazeau, "Evaluating Trace Aggregation for Performance Visualization of Large Distributed Systems," in *Proceedings of the 2014 IEEE International Symposium on Performance Analysis of Systems and Software*, 2014.

[14] R. Lamarche-Perrin, Y. Demazeau, and J.-M. Vincent, "The Best-partitions Problem: How to Build Meaningful Aggregations," in *Proceedings of the 2013 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2013.

[15] ——, "A Generic Set Partitioning Algorithm with Applications to Hierarchical and Ordered Sets," Max-Planck-Institute for Mathematics in the Sciences, Tech. Rep. 51/2014, 2014.

[16] ——, "Building the Best Macroscopic Representations of Complex Multi-Agent Systems," in *Transactions on Computational Collective Intelligence*, ser. LNCS. Springer-Verlag Berlin, Heidelberg, 2014.

[17] S. Kullback and R. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, 1951.

[18] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 3, 1948.

[19] P. Pons and M. Latapy, "Post-processing hierarchical community structures: Quality improvements and multi-scale view," *Theoretical Computer Science*, vol. 412, no. 8-10, 2011.

[20] B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, and *al.*, "An algorithm for optimal partitioning of data on an interval," *IEEE Signal Processing Letters*, vol. 12, no. 2, 2005.

[21] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, and *al.*, "The NAS parallel benchmarks," in *Proceedings of the SuperComputing Conference*, 1991.