Correlating Sub-Phenomena in Performance Data in the Frequency Domain

Tom Vierjahn1,4,*  Marc-André Hermanns3,4  Bernd Mohr3,4
Matthias S. Müller2,4  Torsten W. Kuhlen1,4  Bernd Hentschel1,4

1 Visual Computing Institute, RWTH Aachen University, Germany
2 Chair for High-Performance Computing, RWTH Aachen University, Germany
3 Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany
4 JARA – High-Performance Computing, Germany

Abstract
Finding and understanding correlated performance behaviour of the individual functions of massively parallel high-performance computing (HPC) applications is a time-consuming task. In this poster, we propose filtered correlation analysis for automatically locating interdependencies in call-path performance profiles. Transforming the data into the frequency domain splits a performance phenomenon into sub-phenomena to be correlated separately. We provide the mathematical framework and an overview over the visualization, and we demonstrate the effectiveness of our technique.

Index Terms: D.2.8 [Software Engineering]: Metrics/Measurement—Performance measures, I.5.4.m [Pattern Recognition]: Applications—Signal processing, I.6.9 [Simulation, Modeling, and Visualization]: Visualization

1 Introduction
Optimizing an application so that it efficiently uses the compute power of a modern high-performance computing (HPC) system requires powerful tools for performance analysis. Insight into correlated performance behaviour is a key part in understanding and thus optimizing the complex behaviour of large-scale simulations. While visual exploration of acquired performance data is a valuable asset to locate correlations, the overall search space is typically large due to the number of performance metrics analysed by modern tools and due to the complexity of modern HPC applications.

Performance visualization is an active research field [1]. However, even major tools like Boxfish [2], VizTorus [3], Cube [4] and ParaProf [5] do not provide automatic correlation analysis. Only the latter allows for manual inspection of correlation among up to four different sections of the performance data.

In order to facilitate instant identification of code regions that interact with and influence one another, we propose to use automatic correlation analysis considering every performance metric and every code region stored in a performance profile. By using the frequency domain, performance phenomena are split into sub-phenomena that can be efficiently analysed separately. That way, newly discovered sub-phenomena can be traced through the data without being obfuscated by already known and understood ones.

2 Performance Profiles, System Topology, Spectra
A call-path profile summarizes an HPC application’s behaviour over a complete analysis run. Performance data is acquired and stored according to specific aspects, the performance metrics \( m \in M \), e.g., execution time, number of function calls issued, or bytes transferred; for the application’s functions in their individual execution contexts, i.e., call paths \( c \in C \), considering caller/callee relationship; for the individual system resources \( s \in S \) that executed the code. Thus, a call-path profile constitutes a mapping \( \nu : M \times C \times S \to \mathbb{R} \), with \( v(m, c, s) \) yielding a severity.

During analysis, by selecting a metric-call-path pair \((m, c)\), analysts specify a severity view \( v_{m,c} \), i.e., a mapping \( v_{m,c} : S \to \mathbb{R} \), such that \( v = \bigcup_{m \in M} \bigcup_{c \in C} v_{m,c} \), with \( v_{m,c}(s) \) yielding the severity for the selected pair \((m, c)\) on a system resource \( s \). The individual severities in such a view can later be visualized in a 3D viewport for detailed examination.

Taking into account the actual compute node, CPU core, and hardwire configuration associated to \( s \in S \), naturally arranges the \( v_{m,c}(s) \) in a 3D Cartesian space \( T \) — the system topology. Fig. 1, left, exemplary shows severities arranged in a 2D system topology.

Further information like the network topology can be used to extent \( T \) meaningfully to an \( n \)-dimensional system topology [6]. Thus, given the dimension sizes \( d_i \in \mathbb{N}, i = 1, \ldots, n \), an injective mapping \( T : T \mapsto S, T = [0, d_1] \times \cdots \times [0, d_n] \) exists, with \( T(x) \) yielding a system resource for each location \( x \) in the system topology \( T \). Consequently, \( v_{m,c}(x) := v_{m,c}(T(x)) \) is used as a shorthand.

Now, \( V_{m,c}(x) \) constitutes a discrete space-domain signal. Thus, a spectrum \( \mathcal{V}_{m,c} : \mathbb{K} \mapsto \mathbb{C} \) in the frequency domain \( \mathbb{K} \subseteq \mathbb{Z}^n \), \( |\mathbb{K}| = |T| \), can be computed in \( O(|\mathcal{X}| \log(\mathcal{X})) \) time [7] using discrete Fourier transform (DFT) \( \mathcal{F} \) via \( \mathcal{V}_{m,c}(\mathcal{X}) = \mathcal{F}(V_{m,c}(x)) \). \( V_{m,c} \) for each non-zero \( v_{m,c} \) is pre-computed. Since \( V_{m,c} \) is Hermitian, i.e., \( V_{m,c}(-k) = V_{m,c}^*(k) \), with \( V_{m,c}^* \) denoting the complex conjugate of \( V_{m,c} \), only \( 0.5 \cdot |\mathbb{K}| + 2^{n-1} \) non-redundant values need to be stored. Fig. 1, second from left, shows the magnitude spectrum \( |V_{m,c}| \), of the exemplary severity view, scaled by 4 so that only integers are shown.

3 Automatic Correlation Analysis
Pearson’s correlation of two \( v_{m_1,c_1}, v_{m_2,c_2} \) in the system topology yields a useful indicator for correlated performance phenomena in \( O(|\mathcal{X}|) \) time. However, that approach has shortcomings: first, it does not detect performance phenomena that are shifted in \( T \); second, already known sub-phenomena may obfuscate new ones. The former can be addressed in \( T \) by using cross correlation, however only in \( O(|\mathcal{X}|^2) \) time. The latter cannot be addressed in \( T \) at all.

Fig. 1 shows an example. The severity view \( v_{m_1,c_1} \) reveals two sub-phenomena: an imbalance along \( x_1 \) with two peaks, and one along \( x_2 \) where the resources associated to every second \( x_2 \) run idle.

Figure 1: Severity view \( v_{m_1,c_1} \) exposing variation, its scaled magnitude spectrum \( 4 \cdot |v_{m_1,c_1}| \), and two other severity views \( v_{m_2,c_2} \) and \( v_{m_3,c_3} \).
The latter resembles an intentional imbalance due to, e.g., using only every second thread in order to utilize CPU caches more efficiently. Considering only the remaining two severity views, Pearson’s correlation yields $r(v_{m_{0},c_{1}},v_{m_{1},c_{1}}) \approx 0.46$ and $r(v_{m_{0},c_{1}},v_{m_{2},c_{2}}) \approx 0.76$. That way, the already known, intentional imbalance obfuscates the presence of the said imbalance along $x_1$ in $v_{m_{1},c_{1}}$.

Transforming the severity views into the frequency domain facilitates efficient use of cross correlation $\ast$ for detecting shifted phenomena and for filtering out known sub-phenomena in $O((|\mathcal{T}|\log|\mathcal{T}|)^\Delta)$ time. Let $\tilde{v}_{m,c}$ denote the mean severity and let $\Delta v_{m,c}(x) = v_{m,c}(x) - \tilde{v}_{m,c}$ denote only the varying part of the severity view $v_{m,c}$. Then, the cross correlation theorem yields $\tilde{s}_{m_{0},c_{0},m_{1},c_{1}}(\Delta x) = v_{m_{0},c_{0}} \ast v_{m_{1},c_{1}} = \mathcal{F}^{-1}[\mathcal{F}(v_{m_{0},c_{0}}) \mathcal{F}(v_{m_{1},c_{1}})](\Delta x)$ for the cross correlation function $g : \mathbb{Z}^n \to \mathbb{R}$ of the severity views $v_{m_{0},c_{0}}, v_{m_{1},c_{1}}$, with $\mathcal{F}^{-1}$ denoting the inverse DFT.

In order to suppress individual sub-phenomena, a filter $W(k) \in [0,1]$ is defined for each axis $i$ of $\mathcal{T}$ such that $\tilde{V}^{\ast}_{m_{0},c_{0}}(k)|\tilde{V}^{\ast}_{m_{1},c_{1}}(k)| = \sum_{i=1}^{n} W^{2}(k) |\tilde{V}^{\ast}_{m_{1},c_{1}}(k)|$.

Cosine weighting

\[ W_i(k \neq 0) = |k_i|^{-1} , \quad W_0(0) = 0, \quad k = [k_1 \ldots k_n]^T \]

is a straightforward choice fulfilling the above constraint, since obviously $\sum_{i=1}^{n} W_i(k \neq 0) = 1$. Let $f_i \in [0,1]$ specify whether the sub-phenomena along axis $i$ shall be considered during correlation analysis, then a filtered cross correlation function $\mathcal{G}$ is derived with

\[ \mathcal{G}_{m_{0},c_{0},m_{1},c_{1}}(\Delta x) = \mathcal{F}^{-1}\left[ \sum_{i=1}^{n} f_i W_i^2(k) \tilde{V}^{\ast}_{m_{0},c_{0}}(k)|\tilde{V}^{\ast}_{m_{1},c_{1}}(k)|(\Delta x) \right]. \]

Let now $R_{\mathcal{G}}(v_{m_{0},c_{0}},v_{m_{1},c_{1}})(\Delta x) = \mathcal{G}_{m_{0},c_{0},m_{1},c_{1}}(\Delta x) / \sqrt{\mathcal{G}_{m_{0},c_{0},m_{0},c_{0}}(\Delta x) \mathcal{G}_{m_{1},c_{1},m_{1},c_{1}}(\Delta x)}$ if both involved $\mathcal{G}_{m_{0},c_{0},m_{0},c_{0}}(\Delta x) \neq 0$, and $R_{\mathcal{G}}(v_{m_{0},c_{0}},v_{m_{1},c_{1}})(\Delta x) = 0$, otherwise. Let furthermore $\Delta x^* = \text{arg max}_{\Delta x} |R_{\mathcal{G}}(\Delta x)|$, then

\[ r_{\mathcal{G}}(v_{m_{0},c_{0}},v_{m_{1},c_{1}}) = R_{\mathcal{G}}(v_{m_{0},c_{0}},v_{m_{1},c_{1}})(\Delta x^*) \in [-1,1] \]

yields the filtered correlation of $v_{m_{0},c_{0}}$ and $v_{m_{1},c_{1}}$. Values close to 1 are likely caused by similar execution behaviour, values close to -1 are likely caused by interdependencies like synchronization.

Using $f_1 = 1$ and $f_2 = 0$ for filtering out the intentional imbalance along $x_2$ in the above example (Fig. 1) yields $r_{\mathcal{G}}(v_{m_{0},c_{0}},v_{m_{1},c_{1}}) \approx 0.9$ and $r_{\mathcal{G}}(v_{m_{0},c_{0}},v_{m_{2},c_{2}}) \approx 0.0$. Thus, our technique ignores the imbalance along $x_2$ as desired whereas the correlation of the imbalance along $x_1$ is correctly detected.

4 Interactive Visualization

The automatic filtered correlation analysis is integrated into an interactive visualization tool. This has been developed according to requirements posed by HPC experts. It is inspired by the Cube performance profile browser [4] and by ParaProf’s topology plots [5].

In the visualization (Fig. 2), each system resource is depicted by a small, bevelled cube, arranged according to the Cartesian system topology. Each system resource’s severity is rendered colour-coded onto the respective cube with a user-defined colour map – green (small) through yellow to red (large) is used as a default.