Correlating Sub-Phenomena in Performance Data in the Frequency Domain

Tom Vierjahn[1,4,*] Marc-André Hermanns[3,4] Bernd Mohr[3,4]
Matthias S. Müller[2,4] Torsten W. Kuhlen[1,4] Bernd Hentschel[1,4]

[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* $m \in \mathcal{M}$, 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* $c \in \mathcal{C}$, considering caller/callee relationship; for the individual *system resources* $s \in \mathcal{S}$ that executed the code. Thus, a call-path profile constitutes a mapping $v : \mathcal{M} \times \mathcal{C} \times \mathcal{S} \mapsto \mathbb{R}$, with $v(m,c,s)$ yielding a *severity*.

*e-mail: vierjahn@vr.rwth-aachen.de

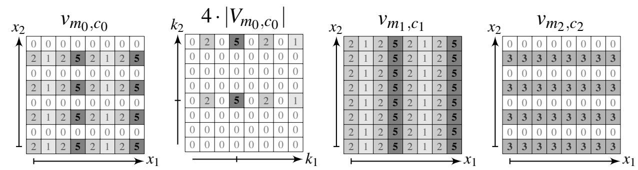

Figure 1: Severity view v_{m_0,c_0} exposing variation, its scaled magnitude spectrum $4 \cdot |V_{m_0,c_0}|$, and two other severity views v_{m_1,c_1} and v_{m_2,c_2}.

During analysis, by selecting a metric-call-path pair (m,c), analysts specify a *severity view* $v_{m,c}$, i.e., a mapping

$$v_{m,c} : \mathcal{S} \mapsto \mathbb{R} \quad \text{, such that} \quad v = \bigcup_{m \in \mathcal{M}} \bigcup_{c \in \mathcal{C}} v_{m,c} \quad ,$$

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 hardware thread associated to an $s \in \mathcal{S}$, naturally arranges the $v_{m,c}(s)$ in a 3D Cartesian space $\mathcal{T}$ – 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 extend $\mathcal{T}$ 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 $T : \mathcal{T} \mapsto \mathcal{S}, \mathcal{T} = [0,d_1) \times \ldots \times [0,d_n)$ exists, with $T(\mathbf{x})$ yielding a system resource for each location $\mathbf{x}$ in the system topology $\mathcal{T}$. Consequently, $v_{m,c}(\mathbf{x}) := v_{m,c}(T(\mathbf{x}))$ is used as a shorthand.

Now, $v_{m,c}(\mathbf{x})$ constitutes a discrete space-domain signal. Thus, a spectrum $V_{m,c} : \mathcal{K} \mapsto \mathbb{C}$ in the frequency domain $\mathcal{K} \subset \mathbb{Z}^n, |\mathcal{K}| = |\mathcal{T}|$, can be computed in $\mathcal{O}(|\mathcal{T}| \log |\mathcal{T}|)$ time [7] using discrete Fourier transform (DFT) $\mathcal{F}$ via $V_{m,c}(\mathbf{k}) = \mathcal{F}[v_{m,c}(\mathbf{x})](\mathbf{k})$. $V_{m,c}$ for each non-zero $v_{m,c}$ is pre-computed. Since $V_{m,c}$ is Hermitian, i.e., $V_{m,c}(-\mathbf{k}) = V^*_{m,c}(\mathbf{k})$, with $V^*_{m,c}$ denoting the complex conjugate of $V_{m,c}$, only $0.5 \cdot |\mathcal{T}| + 2^{n-1}$ 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_0,c_0}, v_{m_i,c_j} in the system topology yields a useful indicator for correlated performance phenomena in $\mathcal{O}(|\mathcal{T}|)$ time. However, that approach has shortcomings: first, it does not detect performance phenomena that are shifted in $\mathcal{T}$; second, already known sub-phenomena may obfuscate new ones. The former can be addressed in $\mathcal{T}$ by using cross correlation, however only in $\mathcal{O}(|\mathcal{T}|^2)$ time. The latter cannot be addressed in $\mathcal{T}$ at all.

Fig. 1 shows an example. The severity view v_{m_0,c_0} 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.

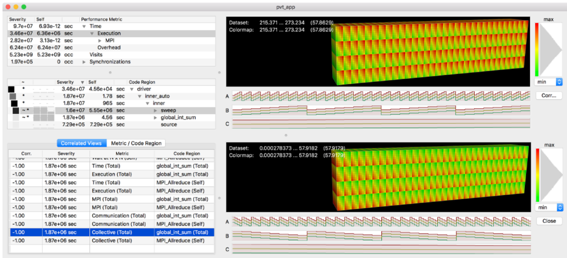

Figure 2: Visualization of two correlated severity views, $r_{\mathbf{f}}(\cdot) = -1$.

The latter resembles an *intentional* imbalance due to, e.g., using only every second thread in order to utilize CPU caches more efficiently. Considering the remaining two severity views, Pearson's correlation yields $r(v_{m_0,c_0}, v_{m_1,c_1}) \approx 0.46$ and $r(v_{m_0,c_0}, v_{m_2,c_2}) \approx 0.76$. 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 $\star$ for detecting shifted phenomena *and* for filtering out known sub-phenomena in $\mathcal{O}(|\mathcal{T}|\log|\mathcal{T}|)$ time. Let $\overline{v}_{m,c}$ denote the mean severity and let $\widetilde{v}_{m,c}(\mathbf{x}) = v_{m,c}(\mathbf{x}) - \overline{v}_{m,c}$ denote only the varying part of the severity view $v_{m,c}$. Then, the cross correlation theorem yields $g_{m_a,c_b,m_y,c_z}(\Delta\mathbf{x}) = \widetilde{v}_{m_a,c_b} \star \widetilde{v}_{m_y,c_z} = \mathcal{F}^{-1}[\widetilde{V}^*_{m_a,c_b}(\mathbf{k})\widetilde{V}_{m_y,c_z}(\mathbf{k})](\Delta\mathbf{x})$ for the cross correlation function $g : \mathbb{Z}^n \to \mathbb{R}$ of the severity views v_{m_a,c_b}, v_{m_y,c_z}, with $\mathcal{F}^{-1}$ denoting the inverse DFT.

In order to suppress individual sub-phenomena, a filter $W_i(\mathbf{k}) \in [0,1]$ is defined for each axis i of $\mathcal{T}$ such that $\widetilde{V}^*_{m_a,c_b}(\mathbf{k})\widetilde{V}_{m_y,c_z}(\mathbf{k}) = \sum_{i=1}^{n} W_i^2(\mathbf{k})V^*_{m_a,c_b}(\mathbf{k})V_{m_y,c_z}(\mathbf{k})$. Cosine weighting

$$W_i(\mathbf{k} \neq \mathbf{0}) = k_i \cdot \|\mathbf{k}\|^{-1} \ , \ W_i(\mathbf{0}) = 0 \ , \ \mathbf{k} = \begin{bmatrix} k_1 & \ldots & k_n \end{bmatrix}^\top$$

is a straightforward choice fulfilling the above constraint, since obviously $\sum_{i=1}^{n} W_i^2(\mathbf{k} \neq \mathbf{0}) = 1$. 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 $g_{\mathbf{f}}$ is derived with

$$g_{\mathbf{f},m_a,c_b,m_y,c_z}(\Delta\mathbf{x}) = \mathcal{F}^{-1}\left[\textstyle\sum_{i=1}^{n} f_i W_i^2(\mathbf{k})V^*_{m_a,c_b}(\mathbf{k})V_{m_y,c_z}(\mathbf{k})\right](\Delta\mathbf{x}) \ .$$

Let now $R_{\mathbf{f}}[v_{m_a,c_b}, v_{m_y,c_z}](\Delta\mathbf{x}) = \frac{g_{\mathbf{f},m_a,c_b,m_y,c_z}(\Delta\mathbf{x})}{\sqrt{g_{\mathbf{f},m_a,c_b,m_a,c_b}(\mathbf{0})g_{\mathbf{f},m_y,c_z,m_y,c_z}(\mathbf{0})}}$ if both involved $g_{\mathbf{f},m_i,c_j,m_i,c_j}(\mathbf{0}) \neq 0$, and $R_{\mathbf{f}}[v_{m_a,c_b}, v_{m_y,c_z}](\Delta\mathbf{x}) = 0$, otherwise. Let furthermore $\Delta\mathbf{x}' = \arg\max_{\Delta\mathbf{x}} R_{\mathbf{f}}[\cdot](\Delta\mathbf{x})$, then

$$r_{\mathbf{f}}(v_{m_0,c_0}, v_{m_1,c_1}) = R_{\mathbf{f}}[v_{m_a,c_b}, v_{m_y,c_z}](\Delta\mathbf{x}') \in [-1,1]$$

yields the filtered correlation of v_{m_a,c_b} and v_{m_y,c_z}. 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 $r_{\mathbf{f}}(v_{m_0,c_0}, v_{m_1,c_1}) \approx 0.9$ and $r_{\mathbf{f}}(v_{m_0,c_0}, v_{m_2,c_2}) = 0.0$. 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.

Table 1: Data Sets, Size, Search Space, Time for Correlation Analysis

Code	Num. Threads	Data Size	Overhead	Num. $v_{m,c} \neq 0$	Correl. Analysis
NEKBONE	1 835 008	8.5 GB	0.6 MB	624	52.3 s
psOpen	65 384	1.0 GB	2.0 MB	2031	1.8 s
Sweep3D	65 384	0.4 GB	53.1 kB	850	0.6 s

Once the analyst has identified a relevant severity view and set suitable $f_1, \ldots, f_n$, correlated severity views are found automatically and presented as a list, odered by $r_{\mathbf{f}}(\cdot)$. When the analyst selects a correlated view from the list, both views are visualized side by side with a synchronized perspective, enabling direct visual comparison. The data in both views can be brushed and filtered to provide better insight into the performance phenomena.

5 Results

The proposed automatic correlation analysis efficiently reduces the size of the search space when looking for correlated severity views by suggesting a list of correlated views. The approach scales well even to performance profiles from massively parallel analysis runs. Tab. 1 lists the overhead for storing $V_{m,c}$ instead of $v_{m,c}$, the number of non-zero severity views, and the time required for finding all views that are correlated to a single $v_{m,c}$.

6 Conclusion

To our knowledge, the proposed system is the first that automatically finds and visualizes correlation in profile-based data. Known performance-phenomena can be suppressed. Our system thus effectively indicates connections between different aspects of an application's performance behaviour. HPC experts rate this ability as a crucial factor in analysing extreme-scale parallel applications. Working in the frequency domain imposes only a negligibly small memory overhead. Correlation analysis has been rated remarkably and sufficiently fast. Although our system aims at analysing performance data, it is very likely that filtered correlation analysis is applicable to other scalar data arranged in Cartesian grids.

Acknowledgements

This work has been partially funded by the German Federal Ministry of Research and Education (BMBF), and by the Excellence Initiative of the German federal and state governments through the Jülich Aachen Research Alliance – High-Performance Computing.

References

[1] K. E. Isaacs, A. Giménez, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz, B. Hamann, and P.-T. Bremer, "State of the Art of Performance Visualization," in *EuroVis - STARs*, 2014.

[2] A. G. Landge, J. A. Levine, A. Bhatele, K. E. Isaacs, T. Gamblin, M. Schulz, S. H. Langer, P.-T. Bremer, and V. Pascucci, "Visualizing network traffic to understand the performance of massively parallel simulations," *IEEE TVCG*, vol. 18, no. 12, pp. 2467–2476, 2012.

[3] L. Theisen, A. Shah, and F. Wolf, "Down to earth: How to visualize traffic on high-dimensional torus networks," in *Proc. 1st Workshop Visual Performance Anal.*, 2014, pp. 17–23.

[4] M. Geimer, P. Saviankou, A. Strube, Z. Szebenyi, F. Wolf, and B. J. N. Wylie, "Further improving the scalability of the Scalasca toolset," in *10th Intl. Conf. Appl. Parallel and Scient. Comput.*, 2012, pp. 463–473.

[5] W. Spear, A. D. Malony, C. W. Lee, S. Biersdorff, and S. Shende, "An approach to creating performance visualizations in a parallel profile analysis tool," in *Euro-Par 2011*, 2012, pp. 156–165.

[6] M. Schulz, J. A. Levine, P.-T. Bremer, T. Gamblin, and V. Pascucci, "Interpreting performance data across intuitive domains," in *Proc. 40th Intl. Conf. Parallel Process.*, 2011, pp. 206–215.

[7] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proc. IEEE*, vol. 93, no. 2, pp. 216–231, 2005.