Idle waves in high-performance computing

Stefano Markidis, Juris Vencels, Ivy Bo Peng, Dana Akhmetova, Erwin Laure, Pierre Henri

PHYSICAL REVIEW E **91**, 013306 (2015)

Idle waves in high-performance computing

Stefano Markidis,[1] Juris Vencels,[1] Ivy Bo Peng,[1] Dana Akhmetova,[1] Erwin Laure,[1] and Pierre Henri[2]
[1]*HPCViz Department, KTH Royal Institute of Technology, Stockholm, Sweden*
[2]*LPC2E–CNRS, Orléans, France*
(Received 10 October 2014; published 20 January 2015)

The vast majority of parallel scientific applications distributes computation among processes that are in a busy state when computing and in an idle state when waiting for information from other processes. We identify the propagation of idle waves through processes in scientific applications with a local information exchange between the two processes. Idle waves are nondispersive and have a phase velocity inversely proportional to the average busy time. The physical mechanism enabling the propagation of idle waves is the local synchronization between two processes due to remote data dependency. This study provides a description of the large number of processes in parallel scientific applications as a continuous medium. This work also is a step towards an understanding of how localized idle periods can affect remote processes, leading to the degradation of global performance in parallel scientific applications.

DOI: 10.1103/PhysRevE.91.013306 PACS number(s): 07.05.Tp, 89.20.Ff, 02.70.−c

I. INTRODUCTION

Since the beginning of distributed computing and the installation of large cluster supercomputers, an unexpected degradation of application performance was observed. One of the most famous examples was observed on the 8192 processor ASCI Q supercomputer at the Los Alamos National Laboratory [1]. A close investigation of the performance degradation showed that system and architecture noise, relatively small idle periods with less than $<1\%$ overhead per process [2], can have a strong impact on the overall performance of the distributed application on the supercomputer [1]. Recent works show that the propagation of these local idle periods in some cases leads to the so-called "noise bottleneck" problem, where an increase of interconnection network speed does not improve the application performance [2]. Future exascale supercomputers will support the parallel execution of billions of processes [3]. Idle periods might affect the whole system, resulting in an ineffective usage of exascale supercomputers. For these reasons, it is of capital importance to understand how an idle period on a single process propagates to other processes. It is well known that relatively short idle periods have a strong impact on the overall performance of the distributed applications on a supercomputer [4]. However, the physical mechanism of how noise propagates and affects all the other processes is still uncertain. In this article, we shed light on the propagation of noise by showing how an idle period propagates to other processes by local synchronization with other processes. By spectral analysis of the processes behavior in a typical distributed scientific application, it is shown that there is a propagation of nondispersive idle waves at phase velocity that is inversely proportional to the average busy period. This is the initial time that a large number of processes in a scientific application is described as a continuous medium and that idle waves in distributed applications have been identified and analyzed.

In scientific applications running on supercomputers, computation is divided among processes, each one solving part of the problem. For instance, scientific applications often solve partial differential equations (PDE) that are first discretized and then solved numerically on a grid. Typically, the domain-decomposition technique is used: the variables defined on a part of the mesh (domain) are assigned to a single process that performs calculations on these data [5]. Often data located on a different process are needed, and therefore communication between processes is performed. Data are communicated either explicitly by sending and receiving messages (message passing) or by directly accessing memory space located on a different process (remote direct memory access) [6]. The single process of the distributed application can be in the idle state when the process is awaiting data from other processes, or it can be in the busy state when it carries out computation or system work. In scientific applications using the message-passing parallel programming model, only busy processes send messages while an idle process becomes busy when it receives a message. In this article, a simple, distributed stencil application whose communication occurs only among nearest-neighbor processes is studied. It is shown how an extended busy period on one process generates idle periods that propagate to other processes as waves. These idle waves are perturbations traveling through processes accompanied by a transfer of information [7].

This article is organized as follows. The second section describes the simulation method used in this work, the third section presents the results, and finally the fourth section summarizes the results and concludes the article.

II. METHODS

The majority of scientific applications solves PDEs on a grid that is distributed over several processes. As an example of such a distributed computing application, the heat equation in one-dimensional Cartesian geometry for the variable u is solved. The equation is discretized by a finite difference in time explicitly with n representing the time level on a computational grid whose grid-cell index is denoted with i as follows:

$$\begin{aligned} &\partial u/\partial t = \alpha\, \partial^2 u/\partial x^2 \\ \Longrightarrow\ & u_i^{n+1} = u_i^n + \alpha\left(u_{i-1}^n - 2u_i^n + u_{i+1}^n\right)\Delta t/\Delta x^2, \quad (1) \\ & i = 1,2,\ldots,M, \end{aligned}$$

where t,x are time and space, α is a constant, Δx and Δt are the grid spacing and time step, respectively. The domain decomposition technique is used: each process calculates Eq. (1) for $N = M/$No. of processes cells. Special care is needed to calculate the boundary values for u_1^n and u_N^n as the u_0^n and u_{N+1}^n values are located on the memory space of different processes [8]. These two values are located on the processes storing information about contiguous domains. These processes are called "nearest-neighbor" processes. Before proceeding to the calculation, messages are exchanged so it is possible to calculate values for u_1^n and u_N^n. This exchange of messages implies an implicit local synchronization with nearest-neighbor processes as each process is waiting for messages from contiguous domains.

The code to solve Eq. (1) in a distributed environment has been implemented in FORTRAN programming language. The Message Passing Interface (MPI) communication library is used to send and receive messages carrying information about the values u_1^n and u_N^n [6]. The code uses blocking send and receive operations: these functions only return and proceed with the execution only after the communication is finished [6]. The busy periods of the distributed computation are obtained by using traces from the FORTRAN code running on the supercomputers: the timing between two different MPI function calls and the number of bytes transmitted for each process have been recorded in files called "traces". The sensitivity error in timing the function call is ± 1 μs. In this work, the LogGOPSim simulator of parallel applications [9] has been used. The busy periods of the application are provided to LogGOPSim simulator that calculates the idle periods by modeling the communication using the LogGOPS network model [9–11]. In this model, the most important parameters to describe the transport of a message from one process to another one are the latency L (the start-up time to send a message), the time cost per byte for sending a message g (gap parameter), and o that reflects the CPU overhead for a communication [9]. In the reference simulations presented here, we use $L = 2.5$ μs and $o = 1.5$ μs. These values mainly depend on the interconnection network in use and vary on different supercomputer architectures. L and o values can be determined experimentally using the Netgauge tool [12]. Typical L and o values for different kind of architectures range between 1.5–10 μs and 0.5–3.5 μs, respectively [2]. The message size in the test application is only 8 bytes (size for a double precision floating-point format), so the communication cost related to g is negligible. The rendezvous protocol that is in use when having small messages is always used to simulate send and receive operations [9]. The traces of the application busy periods are obtained on the KTH Lindgren Cray supercomputer that consists of 1516 computing nodes, each equipped with two 12-core "Magny-Cours" AMD processors and is connected with a Cray Gemini interconnection network. The FORTRAN PGI compiler version 4.0.46 and CRAY MPI implementation, based on MPICH implementation, version 5.5.1 were used in the tests.

III. RESULTS

The perturbation of the busy period is driven by increasing the number of cells to compute on one process at a given computational cycle in the FORTRAN application. The physical mechanism of the generation of idle waves is clear when investigating the plot in Fig. 1. By driving a perturbation on rank four, two idle waves are generated. An extended busy period first generates two idle periods on the nearest-neighbor processes that are waiting for the information of the boundary values. The propagation speed of the idle wave can be intuitively determined by inspecting Fig. 1 and by considering the communication time needed by a driven idle period to travel through processes (identified by a rank number in the plot). For instance, in Fig. 1 the time taken by the idle period to propagate from rank 5 to 7 is equal to the sum of the communication time for sending a message from rank 5 to 6 and from rank 6 to 7 and one busy period on rank 6.

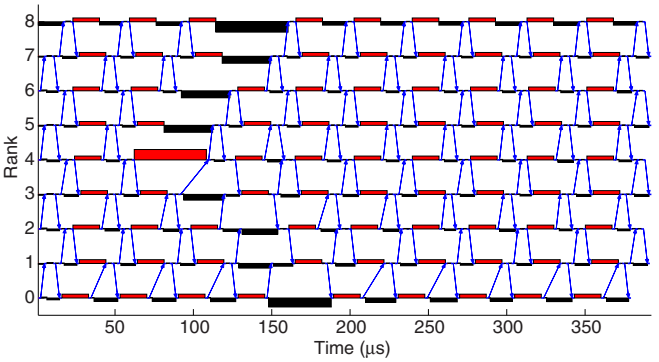

FIG. 1. (Color online) Physical mechanism for the generation of idle waves in a computation distributed over nine processes. Time is presented in the x axis, while the rank of the process is presented in the y axis. The busy periods are represented by the dark gray (red), idle periods in black, and communication in light gray (blue) lines. An extended busy period in the process with rank four generates two idle periods on nearest-neighbor processes (rank three and five). These idle periods propagate to other processes by local synchronization of nearest-neighbor processes as a wave.

In the simulation presented in Fig. 2, a perturbation of the busy period is driven on the process with rank 50 at time $t =$ 110 ms. The average busy period in the simulation is $\langle T_B \rangle =$ 1744.9 μs. $\langle T_B \rangle$ is the average time of the busy period over all the processes. Each process T_B depends on three factors: the computational period, the speed of computing units, and the external (to the application) system operations on the processor. The contour plot in the top panel of Fig. 2 presents the busy and idle times in white and black, respectively. Two propagating idle wave fronts in black are clear in this plot. An insert in the top panel shows an enlargement of the time-rank space with one wave front. The propagation speed for the two idle waves is calculated by measuring the slope of the black line and results in $v_P \simeq \pm 0.0012$ rank/μs. The amplitude of the two idle wave fronts decreases after traveling approximately 30 processes. A spectral analysis of the system allows us to understand the nature of the idle waves and to calculate their phase and group velocities. The numerical dispersion relation is shown in the bottom panel of Fig. 2. It is obtained by first taking a two-dimensional fast Fourier transform of the time-rank signal after applying a Hamming window [13] in time direction (to account for the temporal nonperiodicity) and plotting its absolute value in logarithmic scale [14,15] in

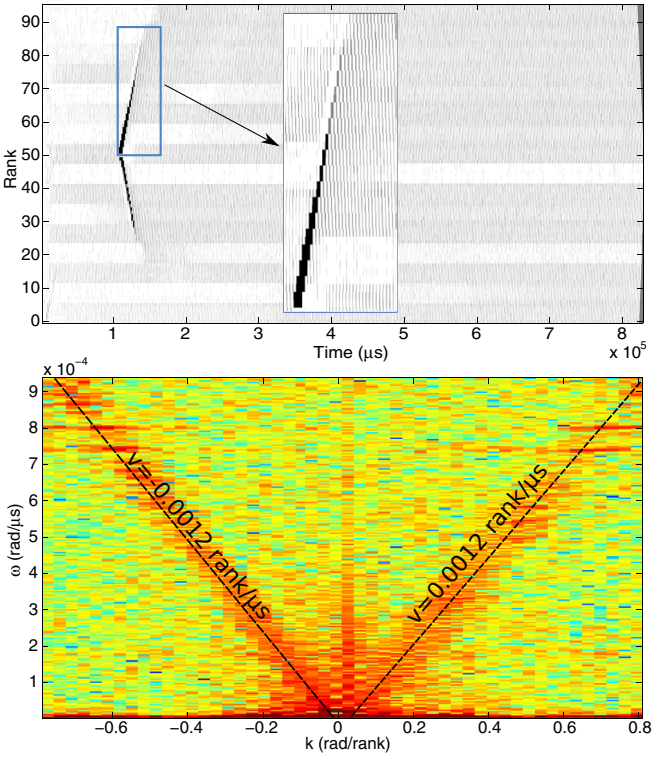

FIG. 2. (Color online) Idle waves in a simulation with 96 processes, $N = 100\,000$ and a driven perturbation of the busy period (five times the average busy period) on process with rank 50 at time $t = 110$ ms. The plot in the top panel presents time on the x axis and process rank on the y axis. The white and black colors indicate the busy and idle periods, respectively. The bottom panel shows the numerical dispersion relation plot obtained by spectral analysis (maximum amplitude in red). Two nondispersive waves with opposite propagation speeds $v_P \simeq \pm 0.0012$ rank/μs are present.

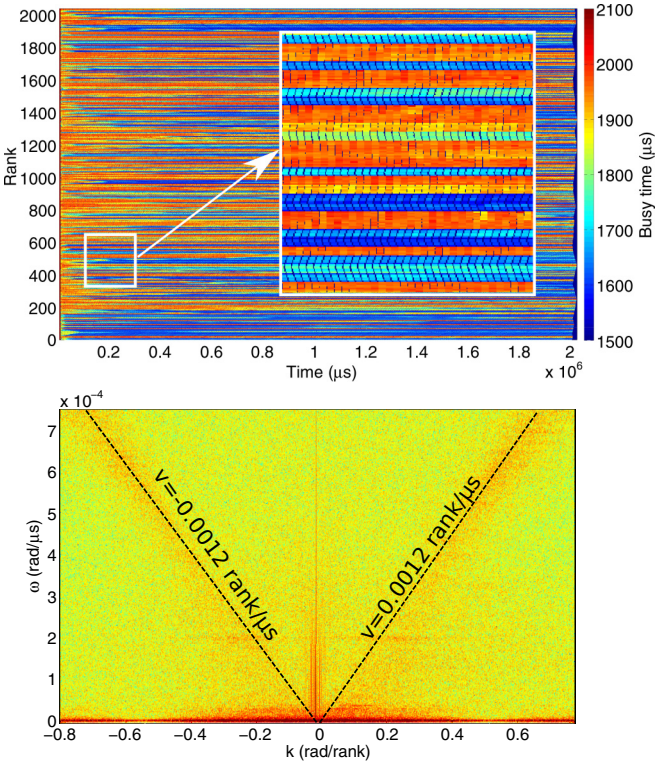

FIG. 3. (Color online) Idle waves in a simulation with 2040 processes with $N = 100\,000$ and without a perturbation. In the top panel, a contour plot of busy periods. The insert presents an enlargement of the contour plot including 168 processes. The bottom panel shows the numerical dispersion relation of idle waves.

angular wave number k-angular frequency ω space. Note that the idle waves are nondispersive. This fact, combined with the observation of the amplitude decrease of the wave front (top panel of Fig. 2), suggests some damping mechanism of the generated idle waves.

It has been found that idle waves are always present in the simulations whether or not a perturbation is driven. In fact, busy periods are different even if the computational workload is the same, i.e., the number of cells per rank is the same. This is because the computing units might be faster or might be used for system operations. Figure 3 shows a contour plot of the busy period in a simulation with 2040 processes without a driven perturbation. The busy periods range between $\Delta T_B =$ 2137–1538 μs with average $\langle T_B \rangle = 1785$ μs and standard deviation $\sigma_{TB} = 139$ μs. Some processes have constant busy periods in time while some other processes have varying busy periods. It can occur that the nearest-neighbor processes have different busy periods, leading to the generation of a sea of idle waves. The bottom panel of Fig. 3 presents the numerical dispersion relation of the idle waves propagating with phase velocity $v_P \simeq \pm 0.0012$ rank/μs.

As is clear in Fig. 1, the propagation speed of idle waves v_P depends on the average busy period of the application $\langle T_B \rangle$ and on the interconnection network parameters, latency L, and the CPU overhead for communication o. A series of simulations with varying $\langle T_B \rangle$, L, and o was carried out to determine the dependency of v_P on $\langle T_B \rangle$, L, and o. The top panel of Fig. 4 shows $1/v_P$ with varying $\langle T_B \rangle$ and constant $L = 2.5$ μs,$o = 1.5$ μs. The linear best fit of the data points is $1/v_P = 0.48\langle T_B \rangle + 15.3$. The bottom panel presents $1/v_P$ with varying L (bottom x axis and black line) and o (top x axis and red line) and constant $\langle T_B \rangle = 152$ μs, $o = 1.5$ μs (for varying L) and $L = 2.5$ μs (for varying o). The linear best fits of the data points for the two cases are $1/v_P = 0.97L + 84.2$ (black line) and $1/v_P = 3.07o + 81.8$ (red line). Several additional experiments confirm the linear dependency of $1/v_P$ on $\langle T_B \rangle$, L, and o with corresponding approximated coefficients 0.5, 1, and 3.0. An equation for the propagation speed of the idle waves has been experimentally determined as

$$v_P \simeq \pm 2/(\langle T_B \rangle + 2L + 6o) \text{ rank/s}. \quad (2)$$

In our simulations, we found that the strong interaction of two driven idle fronts leads to the disappearance of the two driven idle waves, similar to the destructive interference of two waves. This is clear from Fig. 5, where a contour plot of the busy and idle periods in a simulation with periodic boundary conditions on 384 processes is shown in white and black colors, respectively. A perturbation is driven on the process with rank zero and propagates in two opposite directions. The two idle waves interact approximately at time 4 ms and disappear after the destructive interaction. An insert in Fig. 5 presents an

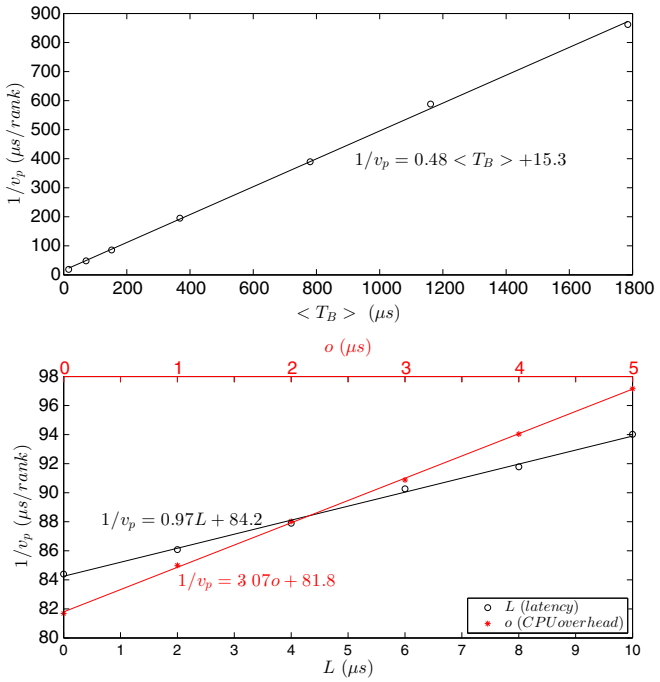

FIG. 4. (Color online) $1/v_P$ dependency on $\langle T_B \rangle$ (top panel), L (bottom panel black line), and o (bottom panel red line). Linear best fits are superimposed to the data points.

enlargement of the two idle wave interactions in the space time-rank.

Collective communication is an operation that involves all processes [16]. In scientific applications, a common use of collective communication is to calculate the inner product of two vectors distributed among different processes. The local (to the process) components of the vectors are first multiplied and summed on a single process. Then the result of the local inner product is communicated from all the processes to one process, which is typically with rank zero (root process) and is added to the results from other processes. Collective communication is achieved by several point-to-point communications

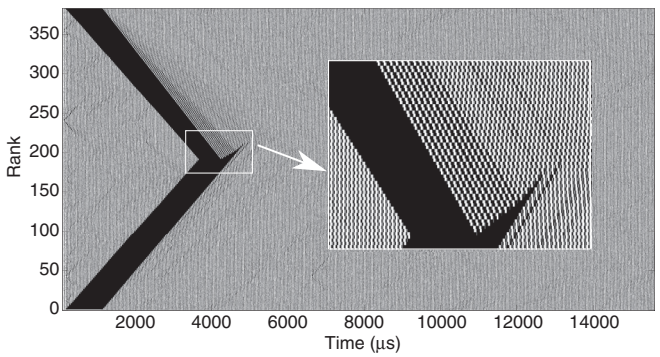

FIG. 5. Interaction between two idle waves in a simulation with 384 processes, $N = 100\,000$, and a driven perturbation of the busy period on rank zero. The plot presents time on the x axis and process rank on the y axis. The white and black colors indicate the busy and idle periods, respectively. The insert in the picture represents an enlargement of the rank-time plot during the idle wave interaction.

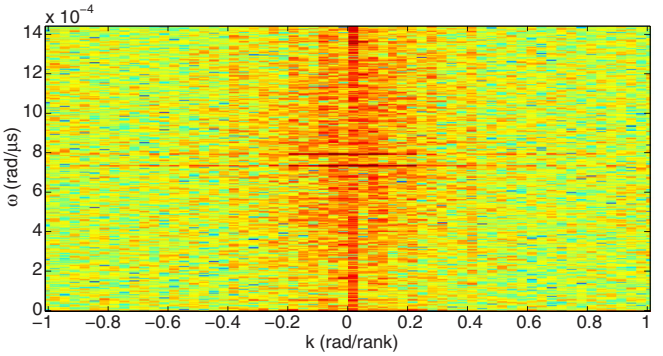

FIG. 6. (Color online) Numerical dispersion relation obtained from a simulation with communication with nearest-neighbor communication and a collective operation at each computational cycle with $N = 100\,000$ on 96 processes.

that do not necessarily involve nearest-neighbor processes. For this reason, idle waves propagating through nearest-neighbor processes cease to propagate in the presence of a collective operation. To prove this, we add a collective operation after the nearest-neighbor communication and the computation of Eq. (1) at each cycle of our application. In this simulation, the reduction operation is modeled with LogGOPSim using the binomial tree algorithm. The numerical spectral analysis of this case is presented in Fig. 6. A clear signature of idle waves propagating through nearest-neighbor communication is not present.

We finally present the results relative to simulations using nonblocking send and receive operations. The nonblocking communication consists of two phases: in the first phase the communication starts and then a completion is enforced in the second phase. This two-stage operation allows us to overlap communication and computation by starting communication first, computing some operations that do not depend on communicated values, and enforcing the completion of communication before finishing computing. In our scientific application, this is achieved by starting communication of the boundary values first, computing the value u_i^{n+1} for

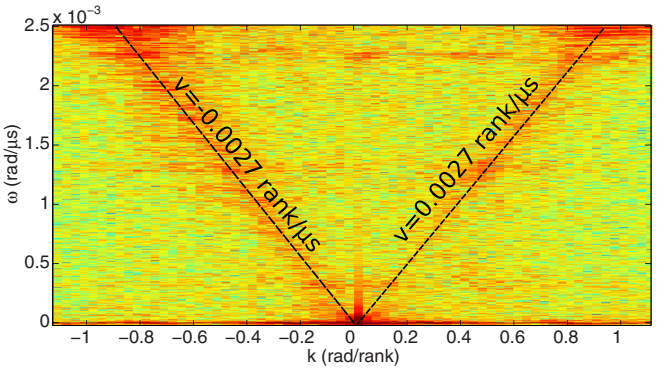

FIG. 7. (Color online) Numerical dispersion relation obtained from a simulation using nonblocking send and receive for communication between nearest-neighbor processes with $N = 100\,000$ on 96 processes. In this case, the propagation speed of idle waves is $v_P \simeq \pm 0.0027$ rank/μs.

$i = 2, \ldots, N-1$, completing the communication, and finally computing u_1^{n+1} and u_N^{n+1}. We remark that synchronization between nearest-neighbor processes is still present when nonblocking communication is used. For this reason, idle waves propagate even in the presence of nonblocking communication with nearest-neighbor processes. Because of the overlap of computation and communication, the propagation speed of idle waves is inversely proportional to the effective average busy period. Figure 7 shows the numerical dispersion relation obtained from a simulation with nonblocking send and receive for communication between nearest-neighbor processes with $N = 100\,000$ on 96 processes. The two idle waves propagating with opposite velocities are still present. Idle waves in nonblocking communication are faster than the idle waves in blocking communication (see Fig. 2) as the effective average busy period is smaller when there is an overlap of communication and computation.

IV. DISCUSSION AND CONCLUSION

This work provides for a description of a large number of processes in a scientific application as a continuous medium and identifies the presence of idle waves propagating through processes via local synchronization. The main results reported in this article are as follows.

(1) Idle periods are generated locally (to the process) as a result of a local busy period unbalance and propagate globally by local synchronization of the processes. In this way, a localized computing unbalance can impact the performance of all the distributed computing processes.

(2) Idle waves are nondispersive in nature and their propagation speed depends on the average busy period and on the interconnection network parameters. If the average busy period is much larger than the time required to send and receive a message, then $v_P \approx 2/\langle T_B \rangle$ rank/s. This speed is relatively slow when compared to the whole execution time of the applications, i.e., in an application with 100 processes it takes 50 computational cycles (with computing time $\langle T_B \rangle$) for an idle wave to travel across all the processes. For this reason, idle waves remain in the system for a relatively long time in the absence of absorption mechanisms.

(3) Idle waves are always present with and without a driven perturbation. This is because computing units have different computing speeds, and system calls might disturb the computation. Perturbations of the busy period occur frequently in scientific applications. The major source of idle waves is the root process as it is often used for I/O activities in combination with other tasks.

(4) Idle waves are damped in time. In addition, we observed that two driven idle fronts strongly interact and that the wave fronts amplitudes are strongly modified (damped) after the interaction.

(5) Collective communications, operations that involve all the processes in the application, lead to the absorption of idle waves propagating through nearest-neighbor communication.

(6) The idle wave propagation mechanism is based on the local synchronization of processes. Idle waves are still present when nonblocking communication operations are used since synchronization remains. In fact, the nonblocking communication consists of two phases: in the first phase the communication starts and then a completion is enforced in the second phase. Nonblocking communication still requires the synchronization of processes allowing the idle waves to travel.

In our simulations, we assume that the interconnection network is fully connected, while network topologies are typically sparsely connected. For this reason, network congestion is not taken into account in the simulation. In addition, a simple nearest-neighbor communication pattern has been studied while large computer codes often exhibit more complicated communication pattern. As part of future work, idle waves in application with more complex communication patterns will be studied.

The experiment dispersion relation of the idle waves obtained from the numerical simulations reads [using v_P given by Eq. (2)]

$$\omega^2 - v_P^2 k^2 = 0. \tag{3}$$

The numerical experiments also suggest the presence of some damping mechanisms (see Fig. 2). This should guide the building of a model that describes the dynamics of the idle periods duration I_p in large parallel computing applications considered as a continuous medium. The dynamics of idle period duration should be described by a damped wave equation of the form

$$\left(\frac{\partial^2}{\partial t^2} + 2\nu \frac{\partial}{\partial t} + v_P^2 \frac{\partial^2}{\partial x^2} \right) I_p = 0, \tag{4}$$

where x is a continuous description of the rank space and ν a damping rate to be characterized. The preparation of such model should be addressed in future works.

This study poses the basis for understanding the implication of local synchronization and the propagation of idle waves. The main result is that a local (to the process) degradation of performance can propagate globally to all the other processes, leading to an overall degradation of the parallel application on the supercomputer. We suggest that many cases of unexpected performance degradation of distributed scientific applications can be explained in terms of idle waves.

ACKNOWLEDGMENTS

This work was funded by the European Commission through the EPiGRAM project (Grant Agreement No. 610598).

[1] F. Petrini, D. Kerbyson, and S. Pakin, *Supercomputing, 2003 ACM/IEEE Conference* (IEEE, New York, 2003), pp. 55–72.

[2] T. Hoefler, T. Schneider, and A. Lumsdaine, *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE, New York, 2010), pp. 1–11.

[3] W. Gropp and M. Snir, Computing in Science & Engineering **15**, 27 (2013).

[4] K. B. Ferreira, P. Bridges, and R. Brightwell, *Proceedings of the 2008 ACM/IEEE conference on Supercomputing* (IEEE, New York, 2008), p. 19.

[5] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker, *Solving Problems on Concurrent Processors. Vol. 1: General Techniques and Regular Problems* (Prentice-Hall, New York, 1988).

[6] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, Vol. 1 (MIT Press, Cambridge, MA, 1999).

[7] R. P. Feynman, R. B. Leighton, and M. Sands, *The Feynman Lectures on Physics, Desktop Edition Volume I* (Basic Books, New York, 2013).

[8] F. B. Kjolstad and M. Snir, *Proceedings of the 2010 Workshop on Parallel Programming Patterns* (ACM, New York, 2010), p. 4.

[9] T. Hoefler, T. Schneider, and A. Lumsdaine, *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing* (ACM, New York, 2010), pp. 597–604.

[10] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures* (ACM, New York, 1995), pp. 95–105.

[11] T. Hoefler, A. Lichei, and W. Rehm, *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International* (IEEE, New York, 2007), pp. 1–8.

[12] T. Hoefler, T. Mehlan, A. Lumsdaine, and W. Rehm, *High Performance Computing and Communications* (Springer, New York, 2007), pp. 659–671.

[13] A. V. Oppenheim, R. W. Schafer, J. R. Buck *et al.*, *Discrete-Time Signal Processing*, Vol. 2 (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[14] G. Llort, M. Casas, H. Servat, K. Huck, J. Gimenez, and J. Labarta, *2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS)* (IEEE, New York, 2011), pp. 332–339.

[15] M. Casas, R. M. Badia, and J. Labarta, International Journal of High Performance Computing Applications **24**, 335 (2010).

[16] N. Chen, R. K. Karmani, A. Shali, B.-Y. Su, and R. Johnson, *Workshop on Parallel Programming Patterns (ParaPLOP)* (University of California, Berkeley, CA, 2009).