Delay Flow Mechanisms on Clusters

Ayesha Afzal, Georg Hager, Gerhard Wellein
Department of High Performance Computing, Friedrich-Alexander University Erlangen-Nürnberg, Germany
{ayesha.afzal,georg.hager,gerhard.wellein}@fau.de

ABSTRACT

Analytic runtime models of distributed-memory applications are often inaccurate because of the wide range of effects that can disturb the regular compute-communicate cycle. Possible sources of disturbance are long-duration delays, fine-grained on-node system noise, variations in network performance, network contention, and application load imbalance. There is to date no comprehensive theory about how delays from different sources travel through a parallel application running on a cluster, or how collective phenomena break the inherent symmetry of the underlying software and hardware. This is partly because of the huge parameter space involved. In this poster, we use synthetic microbenchmarks to highlight three effects that are of importance in this context: propagation of long-term delays, noise-assisted decay of propagating delays, and noise-induced desynchronization of memory-bound applications. Especially the latter leads to surprising insights about performance features of hybrid (MPI+OpenMP) parallel applications.

KEYWORDS

delay propagation; system noise; desynchronization; clusters

1 PROBLEM AND MOTIVATION

Many parallel applications running on clusters using pure MPI or hybrid MPI+OpenMP have an inherent periodicity, where computation phases alternate with communication phases. In a straightforward model, the runtime of such an application is the sum of computation and communication times: $T = T_{\text{exec}} + T_{\text{comm}}$. This model, however, assumes a regularity that is often not observed in application traces even if application load imbalances and network contention can be ruled out. In effect, the measured runtime may be shorter or longer than the prediction (1).[1] Observed phenomenology depends on many factors, such as code characteristics (memory bound vs. compute bound), communication characteristics (periodic vs. open chain process grid, communication patterns and protocols), and system influence (fine-grained noise, coarse-grained one-off delays, network hardware, etc.) (2). We summarize the consequences of these under the term *nonsynchronicity*.

The vast parameter space of nonsynchronicity phenomena necessitates a restriction to simple setups in order to understand the underlying mechanisms. In this work, we investigate three aspects:

(1) *Delay flow across cluster nodes with node-scalable code.* Injected execution delays are observed as they travel through the parallel application under negligible system noise. We derive an analytic expression for the delay propagation speed that takes the relevant communication characteristics into account.
(2) *Noise-assisted decay of propagating delays.* We show that propagating delays can be partially or completely eliminated by fine-grained system noise and observe near-linear increase of decay rate with average noise amplitude.
(3) *Noise-induced desynchronization of memory-bound code.* If multiple MPI processes compete for memory bandwidth, the inherent "lockstep" pattern of parallel code is unstable with respect to system noise. This may even lead to application speedups.

Unless indicated otherwise, experiments were conducted on up to 18 nodes of one leaf switch of the QDR-InfiniBand cluster "Emmy" at Erlangen Regional Computing Center (RRZE).[2] Each node comprises two ten-core Intel "Ivy Bridge" processors. The clock speed was fixed to the base value of 2.2 GHz.

2 BACKGROUND AND RELATED WORK

This work was motivated by a study of idle waves by Markidis et al. [4] and Peng et al. [5]. We extend their coverage in several directions. Hoefler et al. [3, 2] used their LogGOPSim simulator to investigate the influence of system noise on large-scale applications, but it is neither aware of node-level bottlenecks nor does it take the system topology and different kinds of noise characteristics into account. Here we present only data taken on a real cluster, although we intend to develop an appropriate simulator framework.

3 RESULTS

In order to categorize different influence factors appropriately, we introduce a nomenclature that is detailed on the bottom left of the poster: process topology (open chain vs. closed ring), direction of communication (uni- vs. bidirectional), distance of communication (direct neighbor, next-to-next neighbor, etc.), communication protocol (eager vs. rendezvous), communication flavor (blocking vs. nonblocking, split-wait vs. wait-for-all), and the presence of network contention are assigned labels that are used in graph captions.

3.1 Delay flow: basic flavors of delay propagation on a silent system [1]

In this series of experiments, we inject long-duration delays into the execution on one rank (#5) of a purely compute-bound MPI program (executing a series of floating-point divides followed by MPI communication in a time-stepping loop) running with one process per node in a one-dimensional process topology. Depending on the communication parameters, the resulting "idle wave" propagates with different characteristics (3). For instance, eager-mode communication in one direction ($d = +1$) leads to the idle wave traveling to the end of the process chain (and wrapping around in case of a ring topology). Switching to rendezvous protocol, a second wave travels in the opposite direction due to the inherent dependency created by the inter-process handshake. The speed of propagation

[1] Circled numbers refer to specific areas on the poster; see Fig. 1.

[2] https://www.anleitungen.rrze.fau.de/hpc/emmy-cluster/

in ranks/s can be modeled as follows:

$$v_{\text{silent}} = \frac{\sigma d}{T_{\text{exec}} + T_{\text{comm}}},$$

where T_{exec} and T_{comm} are execution and communication time per step, respectively, d is the maximum distance between communicating processes, and

$$\sigma = \begin{cases} 2 & \text{if bi-direct., split-wait, R mode} \\ 1 & \text{else} \end{cases}.$$

3.2 Interaction and damping of propagating delays

Delays of different duration injected on equidistant MPI ranks (closed ring topology, eager mode) travel through the system and partially cancel each other (5). This effect has been known for some time [4], but it leads to the immediate conclusion that fine-grained noise can interact with propagating delays and slow down the speed of their trailing edge, effectively reducing their duration. The leading edge should be unaffected. Hence, fine-grained noise is expected to dampen propagating idle waves. To test this, we injected statistical, exponentially distributed noise, i.e., execution delays with the same workload as the actual program execution, of varying average duration (between 0% and 25% of the compute time) and tracked the average decay rate of a propagating delay. We found empirically that there is a near-linear relationship between the average noise amplitude and the observed decay rate. We ran the same analysis on a Broadwell-based Omni-Path cluster[3] and on the LogGOPSim simulator [2] (with communication parameters set to the values as measured on the IB cluster), and roughly the same dependence was found.

In other words, slight execution imbalance or variations coming from other sources can effectively lessen the impact of long-duration one-off delays. There is as of now no analytical model for the observed linear dependence [1].

Note that the decay phenomenon is not restricted to noise in the execution phase of the program. An experiment with an MPI code that communicates very large messages (3 MB) to multiple neighbors shows essentially the same result (4): In this case, the noise does not emerge from variations in execution time but from different communication times T_{comm} due to the inherent nondeterminism of a contended network. The effect on the propagating delay is the same.

3.3 Noise-assisted desynchronization and structure formation

All previous experiments and models pertained to core-scalable code, i.e., code that does not address bandwidth bottlenecks on the socket level. In this setting, the "natural" node-level system noise could only induce visible effects over long time periods. If a memory bandwidth bottleneck applies, however, system noise and variations in communication characteristics have a much stronger effect on applications.

We conducted experiments using a strongly memory-bound code (STREAM triad loop), running successively more processes per socket (PPS) and observing the propagation of a strong injected delay with closed ring topology and bi-directional next-neighbor eager mode communication with small messages (6). As soon as memory bandwidth saturation sets in (at ≈4 processes per socket), the reaction of the system to an injected one-off delay changes qualitatively:

(1) The idle wave starts to decay as it travels, and the decay rate is stronger with more processes per socket.
(2) A *desynchronization* effect sets in, where cores show increased idle time, alternating with phases of activity. Since, at any given time, not all cores are active on the memory interface, the available bandwidth per core is larger than in the initial lock-step state.

These observations lead to the following hypothesis: If the application code is strongly memory bound, the parallel code may shift from a perfectly synchronized state, where pure execution and pure communication alternate in lockstep on all processes, to a desynchronized state where part of the processes on each socket execute, utilizing all available memory bandwidth, while others wait or communicate. This would lead to an automatic overlap of communication and computation that was not originally part of the application design.

If the communication overhead is small, the transition into this state can take a long time unless the system is "kicked" via an injected idle period. For larger communication overhead, the transition happens quickly and requires only the natural system noise. To study this in a simplified setting, we ran the code on four sockets in a hybrid setup with five threads per MPI rank and two ranks per socket (7). This way, one MPI rank can already saturate the memory interface of a socket on its own. After already a few time steps, desynchronization sets in and the system stays in this state: Process pairs sharing a socket alternate in utilizing the memory interface. Even though overall performance is better than in the synchronized mode, the communication and waiting time per process will usually be longer.

The steady state has the peculiar property that the "computation front," i.e., the wall-clock time at which each process is at the same simulated time step, forms a wave pattern with a fundamental wavelength equal to the number of ranks.

This transition can only happen when more than one MPI process shares a memory interface; using pure OpenMP loop parallelism on all cores of a node does not leave room for overlap. Also, forcing lockstep by frequent collective operations like `MPI_Reduce` will not allow desynchronization to "slip" into a steady state.

4 CONCLUSION

We have shed light on the propagation properties of one-off delays ("idle periods") injected into MPI applications with a regular compute-communicate pattern. In the absence of significant system noise, the delay propagates with a definite speed that depends on communication characteristics and execution time. We could also show that noise, i.e., fine-grained variations in execution time or communication time, leads to a decay of the idle wave, to the point where it is entirely dissolved: Noisy systems are impervious to idle waves.

In case of memory-bound workloads, the phenomenology becomes a lot more complex. Sharing of memory bandwidth across

[3] https://www.anleitungen.rrze.fau.de/hpc/meggie-cluster/

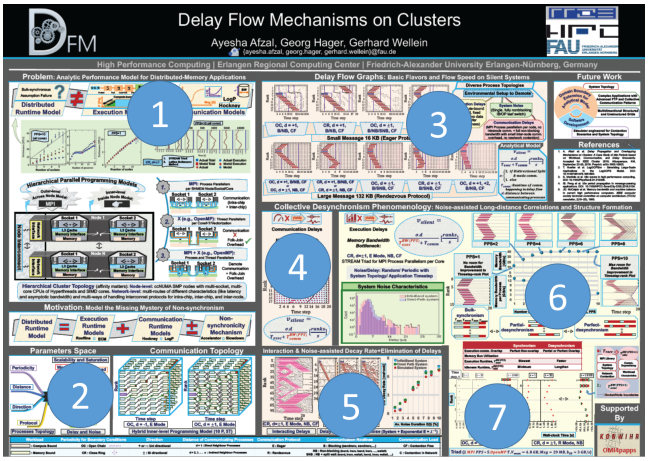

Figure 1: Poster orientation map. Circled numbers mark areas that are referred to in the text of the summary.

MPI processes running within a CPU socket makes the system unstable against small disturbances and ultimately leads to desynchronization and large-scale structure formation. We demonstrated this effect and showed that in case of significant communication overhead, the natural system noise is sufficient to quickly trigger the transition. With small communication overhead, it can be accelerated by an injected one-off delay.

Outlook. Much of the dynamics of idle wave decay and structure formation is still not well understood. We will try to derive a quantitative model of the dependence between idle wave decay rate and noise level. It is also unclear whether and how the particular *statistical* properties of noise impact the decay.

Furthermore, we will investigate the instability of the synchronous phase of memory-bound applications and their seemingly inevitable transition into a phase with desynchronized contended processes.

In order to be able to study a wider parameter range, the development of a versatile simulation tool will be pushed forward. Such a tool should not only be flexible enough to accommodate different noise characteristics but also comprise realistic node-level execution and communication performance models.

Finally, although the simplistic experimental setups used so far led to interesting insights, more complex, real-world applications with advanced point-to-point and collective communication patterns will be investigated.

ACKNOWLEDGMENTS

This work is supported by KONWIHR, the Bavarian Competence Network for Scientific High Performance Computing in Bavaria, under the project name “OMI4papps.” We thank Michael Meier and Thomas Zeiser (RRZE) for excellent technical support.

REFERENCES

[1] A. Afzal, G. Hager, and G. Wellein. Propagation and decay of injected one-off delays on clusters: A case study. *CoRR*, abs/1905.10603, 2019. arXiv: 1905.10603. URL: http://arxiv.org/abs/1905.10603. Accepted for IEEE Cluster 2019.

[2] T. Hoefler, T. Schneider, and A. Lumsdaine. LogGOPSim – Simulating Large-Scale Applications in the LogGOPS Model. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 597–604, Chicago, Illinois. ACM, June 2010. ISBN: 978-1-60558-942-8.

[3] T. Hoefler, T. Schneider, and A. Lumsdaine. Characterizing the influence of system noise on large-scale applications by simulation. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE Computer Society, 2010.

[4] S. Markidis, J. Vencels, I. B. Peng, D. Akhmetova, E. Laure, and P. Henri. Idle waves in high-performance computing. *Physical Review E*, 91(1):013306, 2015.

[5] I. B. Peng, S. Markidis, E. Laure, G. Kestor, and R. Gioiosa. Idle period propagation in message-passing applications. In *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*, pages 937–944. IEEE, 2016.