Distributed Algorithm of Data Allocation in the Fragmented Programming System LuNA

Victor E. Malyshkin[123], Vladislav A. Perepelkin[12], and Georgy A. Schukin[13]

[1] Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of Russian Academy of Sciences, Novosibirsk, Russia
{malysh,perepelkin,schukin}@ssd.sscc.ru
[2] Novosibirsk State National Research University, Novosibirsk, Russia
[3] Novosibirsk State Technical University, Novosibirsk, Russia

Abstract. The paper presents distributed algorithm with local communications Rope-of-Beads for static and dynamic data allocation in the LuNA fragmented programming system. LuNA is intended for implementation of large-scale numerical models on multicomputers with large number of processors and various network topologies. The algorithm takes into account the structure of a numerical model, provides static and dynamic load balancing and can be used in various network topologies.

Keywords: dynamic data allocation, distributed algorithms with local interactions, fragmented programming technology, fragmented programming system LuNA

1 Introduction

Implementation of large-scale numerical models on supercomputers is a challenging problem in high-performance parallel computing. In the light of growing size of supercomputers (in terms of memory capacity, number of cores, etc.) new system algorithms are to be developed for data processing and computations' organization, because to achieve good efficiency and scalability of parallel programs one has to provide its dynamic properties, such as dynamic load balancing, effective resources allocation strategy, etc. So, the complexity of application parallel programming becomes comparable to the one of system parallel programming. To simplify creation of parallel programs, enabled to achieve good performance, LuNA system was developed, that provides automatic parallel program generation [1–4].

In LuNA a program is assembled out of fragments of data and computations on these data. Each fragment of computation (CF) can be viewed as independent process, computing output data fragments (DF) from its inputs. Each DF is single assigned and each CF is executed only once. Fragmented structure is preserved during execution, which allows fragments to migrate between processing elements (PEs) of multicomputer and be executed in parallel.

The quality of parallel programs, generated by LuNA, heavily depends on the quality of resources distribution. In the paper the authors propose a distributed local algorithm of dynamic resources allocation, employed in LuNA system.

2 Related Works

The problem of efficient and scalable data allocation is actively researched.

Worth mentioning are scalable diffusion-like algorithms ([6–8]), since they do not require global interactions, but they lack concerning data structures, balancing speed and allow global imbalance with low load gradient.

Data allocation problems are common in distributed databases ([12–15]) and cloud services ([16–18]). Due to relative small count of objects to distribute and low migration rates, these systems use centralized algorithms, which, because of potentially unlimited number of DFs and high migration rates, are not suitable for large distributed multicomputers.

Good efficiency can be achieved for allocating data of particular structures, such as meshes ([19, 20]). However, these algorithms have very limited application domain.

Worth mentioning are static analysis algorithms, employed in compilers ([9–11]). Their limitation is static decision making at compile time.

Algorithms in [21, 22] do not take data structure into account and do not solve the problem of data search. In [15] a relatively scalable algorithm is presented, but it employs global communications, which should be avoided.

3 Requirements for Data Allocation Algorithm

In order to provide high efficiency and scalability, data allocation algorithm should meet the following requirements:

- To provide equal load of available PEs (static and dynamic load balancing)
- To reduce communications length by taking into account the structure of data
- To tune to behavior of phenomena being modeled
- To be decentralized and use mostly local communications

4 Distributed Algorithms of Data Allocation

Two distributed algorithms of data allocation were developed: Hash-and-Track (HaT) and Rope-of-Beads (RoB), RoB being an improvement over HaT. Next sections present description and comparison of these algorithms.

4.1 Hash-and-Track algorithm

The basic idea of the algorithm is that each PE is responsible for tracking actual location of a subset of DFs. Each DF is tracked by exactly one PE (called *tracker PE*), defined by static hash-function on DF identifier and thus known to all other PEs.

Whenever a DF is created or transferred to another PE, the tracker PE is notified about its new location. If a DF is required on some PE, the tracker PE is queried first and then forwards the request to the actual location of the DF. Thus fixed (three in the worst case) number of interactions is required to obtain any DF from any PE.

The main advantage of the HaT algorithm is its scalability in terms of computational load and extra storage usage. Given the hash function provides uniform distribution of values among PEs, the number of DFs to track will be nearly equal for all PEs.

Drawback of the algorithm is extra non-local communications with tracker PEs, which impede scalability (this is confirmed in "Experiments" section).

To overcome the shortcomings of HaT algorithm, RoB algorithm was developed.

4.2 Rope-of-Beads algorithm

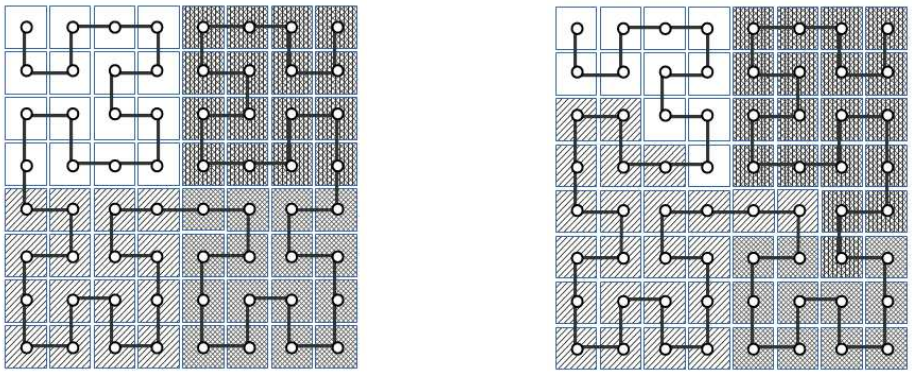
In the RoB algorithm each DF is statically mapped onto a line segment $[0, 1]$ (like beads on a rope – hence the name of the algorithm) and is assigned a real number called *coordinate*. The segment $[0, 1]$ is divided into sub-segments, one for each PE, adjacent sub-segments are mapped to neighbouring (connected by physical link) PEs, thus creating a line of PEs. In such a way, each DF is mapped to a PE. For each DF each PE can compute its coordinate and, if it belongs to its sub-segment, access required DF, or, depending on the value of the coordinate, forward query to a next or previous PE in the line.

In many problems spatially close data elements in domain are related by data dependencies. One good way to construct mapping from multi-dimensional data domain to [0, 1] segment, which preserves locality of DFs, is to use space-filling curves (SFC), for example, Hilbert curve ([23, 24]).

Distribution of DFs with Hilbert curve on 4 PEs is shown on Fig. 1. Fig. 1a shows distribution of equally sized DFs among similar PEs, whereas Fig. 1b shows distribution in a case of non-equally sized DFs and/or PEs having different processing capabilities. In both cases uniform load of PEs is achieved.

The RoB algorithm has the following advantages:

- All communications are local, memory consumption is constant and computational overhead is constant.
- By usage of SFC spatially close DFs will be allocated on the same or neighbor PEs.
- Load balancing is done by dynamic shift of the boundaries of sub-segments and migrating DFs between adjacent PEs. Diffusion-like algorithms may be used for load balancing.



(a) (b)

Fig. 1: Usage of a Hilbert curve for domain decomposition

Drawbacks of the algorithm are:

- DFs structure is reduced to one dimension, therefore some potential locality of the problem may not be utilized.
- The count of hops to find a DF is linear function of a number of PEs in the worst case. However, the search distance is expected to be mostly equal to 0 or 1 for applications with good locality.
- For problems with statically unknown number of DFs, irregular domain structure or dynamically changing data locality it isn't always possible to compute good DFs mapping to $[0, 1]$ segment statically. Therefore, communication overhead may increase.

5 Experiments

To compare performance of the algorithms, LuNA implementation of a solver of Poisson equation with an explicit finite-difference scheme on a regular 3D mesh ([5]) was chosen.

Experiments were conducted on the cluster of Siberian Supercomputing Center with quad core Intel Xeon 5540 processors. The test contained 100 iterations of the solver on $400 \times 400 \times 400$ mesh, subdivided into 16 fragments per each of two decomposition dimensions.

For both algorithms network traffic and total execution time were monitored. Hilbert curve and generic hash function were used for DF and CF distribution.

5.1 Experiment results

Table 1 shows execution times for the both algorithms on different number of PEs. HaT algorithm with hash function for distribution shows the worst results.

Table 1: Execution times (sec.) for the Poisson solver

Number of PEs	1	2	4	8	16
HaT (hash)	214.5	1297.2	2360.4	2400.7	2568.7
HaT (Hilbert)	187.9	101.9	54.5	32.7	17.2
RoB (Hilbert)	175.6	95.6	44.8	26.4	15.2

Usage of Hilbert curve greatly improved execution times. RoB algorithm showed the best results in the majority of cases.

6 Conclusion

The problematics of data distribution automation for implementation of large-scale numerical models for supercomputers is considered. The RoB algorithm for dynamic data allocation for LuNA fragmented programming system is proposed. Performance tests of the RoB algorithm are presented.

Acknowledgments. This work was supported by Russian Foundation for Basic Research (grants 14-07-00381 a and 14-01-31328 mol_a).

References

1. Malyshkin, V.E., Perepelkin, V.A.: LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem. In: PaCT 2011, LNCS, vol. 6873, pp. 53–61. Springer, Heidelberg (2011)
2. Malyshkin, V.E., Perepelkin, V.A.: Optimization Methods of Parallel Execution of Numerical Programs in the LuNA Fragmented Programming System. J. Supercomputing, vol. 61, no. 1, pp. 235–248 (2012)
3. Malyshkin, V.E., Perepelkin V.A.: The PIC Implementation in LuNA System of Fragmented Programming. J. Supercomputing, vol. 69, no. 1, pp. 89–97 (2014)
4. Kraeva, M.A., Malyshkin, V.E.: Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers. J. Future Generation Computer Systems, vol. 17, no. 6, pp. 755–765 (2001)
5. Kireev, S.E., Malyshkin V.E.: Fragmentation of Numerical Algorithms for Parallel Subroutines Library. J. Supercomputing, vol. 57, no. 2, pp. 161–171 (2011)
6. Kraeva, M.A., Malyshkin, V.E.: Dynamic Load Balancing Algorithms for Implementation of PIC Method on MIMD Multicomputers. J. Programmirovanie, no. 1, pp. 47–53 (in Russian) (1999)
7. Hu, Y.F., Blake, R.J.: An Improved Diffusion Algorithm for Dynamic Load Balancing. J. Parallel Computing, vol. 25, no. 4, pp. 417–444 (1999)
8. Corradi, A., Leonardi, L., Zambonelli F.: Performance Comparison of Load Balancing Policies Based on a Diffusion Scheme. In: Euro-Par'97 Parallel Processing, pp. 882–886. Springer, Heidelberg (1997)
9. Anderson, J.M., Lam, M.S.: Global Optimizations for Parallelism and Locality on Scalable Parallel Machines. In: ACM-SIGPLAN PLDI'93, pp. 112–125. ACM New York, USA (1993)

10. Li, J., Chen, M.: The Data Alignment Phase in Compiling Programs for Distributed-Memory Machines. J. Parallel and Distributed Computing, vol. 13, no. 2, pp. 213–221 (1991)
11. Lee P.: Efficient Algorithms for Data Distribution on Distributed Memory Parallel Computers. J. IEEE Transactions on Parallel and Distributed Systems, vol. 8, no. 8, pp. 825–839 (1997)
12. Yu-Kwong Kwok, Ahmad, I.: Design and Evaluation of Data Allocation Algorithms for Distributed Multimedia Database Systems. IEEE Journal on Selected Areas in Communications, vol. 14, no. 7, pp. 1332–1348. IEEE (1997)
13. Iacob, N.M.: Fragmentation and Data Allocation in the Distributed Environments. Annals of the University of Craiova - Mathematics and Computer Science Series, vol. 38, no. 3, pp. 76–83 (2011)
14. Jagannatha, S., Geetha, D.E., Suresh Kumar, T.V., Rajani Kanth, K.: Load Balancing in Distributed Database System using Resource Allocation Approach. J. Advanced Research in Computer and Communication Engineering, vol. 2, no. 7, pp. 2529–2535 (2013)
15. Honicky, R.J., Miller E.L.: Replication Under Scalable Hashing: A Family of Algorithms for Scalable Decentralized Data Distribution. In: 18th International Parallel and Distributed Processing Symposium (2004)
16. Alicherry, M., Lakshman, T.V.: Network Aware Resource Allocation in Distributed Clouds. In: INFOCOM 2012, pp. 963–971. IEEE (2012)
17. AuYoung, A., Chun, B.N., Snoeren, A.C., Vahdat, A.: Resource Allocation in Federated Distributed Computing Infrastructures. In: First Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure (2004)
18. Raman, R., Livny M., Solomon, M.: Matchmaking: Distributed Resource Management for High Throughput Computing. J. Cluster Computing, vol. 2, no. 1, pp. 129–138 (1999)
19. Reddy, C., Bondfhugula, U.: Effective Automatic Computation Placement and Data Allocation for Parallelization of Regular Programs. In: 28th ACM International Conference on Supercomputing, pp. 13–22. ACM New York, USA (2014)
20. Baden, S.B., Shalit, D.: Performance Tradeoffs in Multi-tier Formulation of a Finite Difference Method. In: ICCS 2001, LNCS, vol. 2073, pp. 785–794. Springer, Heidelberg (2001)
21. Ken-ichiro Ishikawa. ASURA: Scalable and Uniform Data Distribution Algorithm for Storage Clusters. Computing Research Repository, abs/1309.7720 (2013)
22. Chawla, A., Reed B., Juhnke, K., Syed, G.: Semantics of Caching with SPOCA: A Stateless, Proportional, Optimally-Consistent Addressing Algorithm. In: USENIX Annual Technical Conference 2011, pp. 33–33. USENIX Association (2011)
23. Lowder, J.K., King, P.J.H.: Using Space-Filling Curves for Multi-dimensional Indexing. In: Advances in Databases, LNCS, vol. 1832, pp. 20–35. Springer, Heidelberg (2000)
24. Moon, B., Jagadish, H.V., Faloutsos, C., Saltz, J.H.: Analysis of the Clustering Properties of the Hilbert Space-Filling Curve. J. IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 1, pp. 124–141 (2001)