Energy and Communication Efficient Partitioning for Large-scale Finite Element Computations

Milinda Fernando
School of Computing
University of Utah,USA

Dmitry Duplyakin
Department of Computer Science
University of Colorado Boulder, USA

Hari Sundar
School of Computing
University of Utah, USA

I. Introduction

Load balancing and partitioning are critical when it comes to parallel computations. Generally partitioning involves equally dividing the work and data among the processors, reducing processor idle time and communication costs. As we march towards exascale machines , the cost of data movement and load-imbalances therein are a major bottleneck for achieving scalability. Space Filling Curves (SFC) are commonly used for partitioning meshes for large-scale PDE systems. They are particularly common while dealing with adaptive meshes, such as those based on quadtrees and octrees. We propose an alternative SFC-based partitioning scheme where we allow some (user-specified) flexibility (tolerance or slack) in the work assignment, so as to minimize the data-dependencies across partitions. Effectively, we allow the flexibility in minimizing the communication-imbalance and overall energy consumption at the cost of a marginal increase in work load-imbalance. The traditional SFC-based partitioning can be recovered by setting the flexibility to zero.

One of the main advantage of SFC based partitioning is the preservation of geometric locality of objects between processors. Depending on the SFC (i.e. Morton, Moore, Hilbert) that used for partitioning, the amount of locality preserved differs [1]. In this work we evaluate how the commonly used SFC's, Hilbert and Morton based partitioning behave with the proposed user specified flexibility in terms of reducing communication costs and overall energy consumption.

A. Flexibility in SFC based partitioning schemes

While partioning adaptive meshes, SFC-based partitioning algorithms are likely to partition using the finest level octant, resulting is increased boundary surface (see Fig 1), as the primary criterion is to equally divide the work (octants) amongst the processes. Our hypothesis is that, it should be possible to find a partition in close proximity to the optimally load-balanced partition, that has a lower inter-process boundary surface. This will enable users to get the partition with the minimum boundary surface, by specifying a tolerance on the load-balance. By incorporating the partitioning step with a top-down SFC ordering, we are able to determine partitions corresponding to coarser (higher in the tree) levels, as long as the load imbalance is within the specified tolerance. This enables us to get the least possible communication within the specified load-imbalance tolerance.

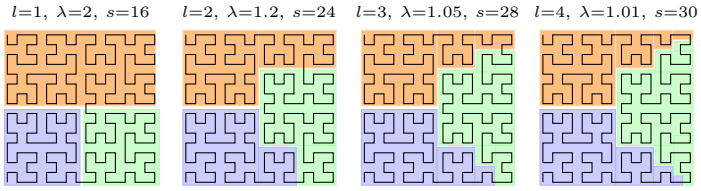

Fig. 1: *Illustration of the increase in communication costs with low tolerance (ideal load balancing). Partitions for the case of $p = 3$ are drawn with the boundary of the partition (s) and the load-imbalance (λ) given along with the level (l) at which the partition is defined. At each level, the orange partition (■) gets the extra load that is progressively reduced. The green partition (■) gets the largest boundary that progressively increases.*

II. Evaluation Methodology

A. Finite element computations & MATVEC

Since our evaluation of the quality of the partition will be based on the evaluation of a MATVEC with the global finite element 'stiffness' matrix, we will give a brief summary of the approach and the assumptions. Please refer to [5], [2] for additional details on performing distributed finite element computations using octree meshes. The MATVEC refers to a function that takes a vector and returns another vector, the result of applying the discretized PDE operator to the the input vector. As the building block for most large-scale PDE solvers, the cost, efficiency and scalability of the MATVEC operation will determine the efficiency and scalability of the overall system. We first loop over the elements in the interior of the processor domain since these elements do not share any nodes with the neighbouring processors. While we perform this computation we communicate the values of the ghost nodes in the background. At the end of the first loop, we use the ghost values from other processors and loop over the remaining elements. These ghosts have to be exchanged after (or before) each MATVEC and therefore characterizes the communication structure of finite element computations.

B. Energy & MATVEC

In order to reason quantitatively about the MATVEC energy consumption, we configured an 8-node compute cluster on CloudLab[3]. The provisioned dedicated resources include physical machines with Intel E5-2630 v3 8-core Haswell CPUs (2.40 GHz) and 128GB ECC Memory interconnected with a 10Gb Ethernet network. This cluster was configured using

the Chef configuration management system as described in [4]. We reused the existing Chef cookbooks ("bundles" of installation and configuration scripts), updated and expanded them, making the process of turning a CloudLab experiment into a compute cluster more streamlined and repeatable.

We used the created cluster to run a set of selected compute jobs. This set contained over 70 8-node jobs summing up to over 7M core-hours of compute time. We scheduled and ran those jobs using the SLURM scheduler. During execution, we obtained on-board IPMI sensor information and recorded every machines instantaneous power draw (in Watts) every second. After job completion, we used the recorded power traces to obtain per-job energy consumption estimates (in Joules). In addition to the total job consumption, we estimated the amount of energy consumed during the MATVEC phase of the computation. In order to eliminate the impact of the dynamic CPU frequency scaling on our energy estimates, we disabled the dynamic scaling and set all CPU cores to run at 2.4 GHz.

III. Results

As mentioned earlier, efficiency in FEM computations mainly relies on MATVEC , we use MATVEC operation as a benchmark to evaluate the quality of the flexible partitions in terms of reducing the overall communication cost & energy consumption. Since we need to deal with ghost octants, each process need to communicate with subset of processes while performing MATVEC operation. Hence for the entire MATVEC operation we can define a communication matrix $\mathcal{M}$, where

$$\mathcal{M} = \begin{cases} m_{ij} & \text{if rank } i \text{ exchange } m_{ij} \text{ data with rank } j \\ 0 & \text{if rank } i \text{ is not exchange data with rank } j \end{cases}$$

Figure 2 shows how the number of non-zero elements (nnz) varies for the same input with different tolerance values. This implies that by introducing flexibility into partitioning we can reduce the overall communication cost associated with MATVEC operation. Reduction of overall communication cost helps towards lower energy consumption (see Fig. 4). In figure 4, default partitioning refers to nearly optimal load(work) balance (with a tolerance of 10^{-5}). As can be seen, the proposed flexible partitioning reduces the energy consumption for both curves, with significantly greater reduction for Hilbert.

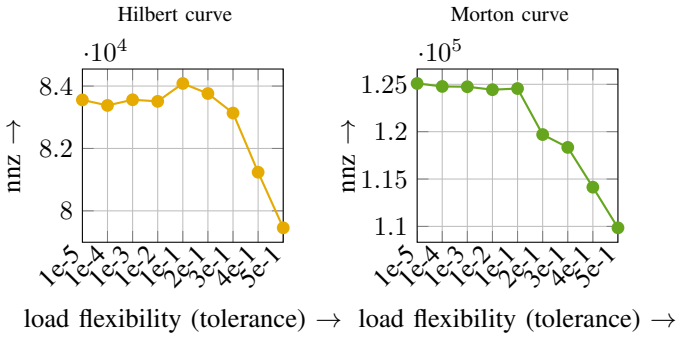

Fig. 2: *Comparison for number of non-zeros (nnz) elements in the communication matrix corresponding to perform* MATVEC *operation based on* HILBERT *and* MORTON *based partitioning schemes for a mesh size of 1B nodes with 4096 mpi tasks with varying tolerance values in TACC's Stampede. Note that the scale difference between the axes in the plots, and for both partitioning schemes we can reduce the nnz (overall communication cost) by increasing the tolerance value.*

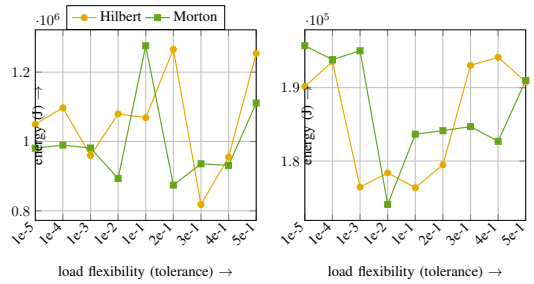

Fig. 3: *Comparison for* MATVEC *energy consumption based on* HILBERT *and* MORTON *based partitioning schemes for a mesh size of 95M & 2M nodes with 256 mpi tasks (left) & 8 mpi tasks (right) with varying tolerance values in CloudLab. Note that the two scenarios depicts the effect of running 32 tasks in a single node Vs. 1 task in a single node.*

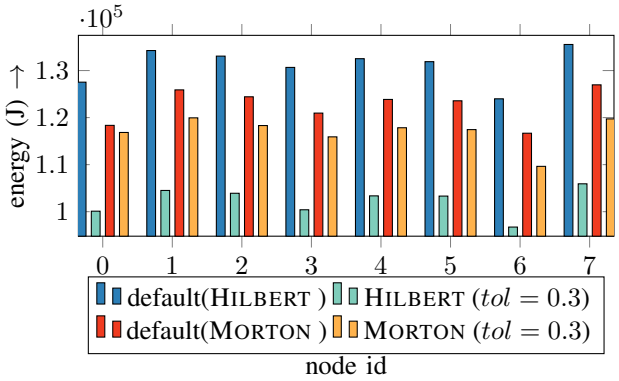

Fig. 4: *Energy consumed by each node while performing* MATVEC *operation , with ideal load balancing (for both* HILBERT *and* MORTON *) Vs. flexible load balancing with a tolerance of* 0.3 *for 95M mesh nodes with 256 mpi tasks in CloudLab8 node cluster.*

References

[1] M. Bader. *Space-filling curves: an introduction with applications in scientific computing*, volume 9. Springer Science & Business Media, 2012.

[2] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[3] CloudLab. Cloudlab cluster. https://cloudlab.us/. Accessed: 2015-11-19.

[4] D. Duplyakin and R. Ricci. Introducing configuration management capabilities into CloudLab experiments. In *Proceedings of the International Workshop on Computer and Networking Experimental Research Using Testbeds (CNERT)*, Apr. 2016.

[5] H. Sundar, R. S. Sampath, S. S. Adavani, C. Davatzikos, and G. Biros. Low-constant parallel algorithms for finite element simulations using linear octrees. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, page 25. ACM, 2007.