Exdasy – A User-Friendly and Extendable Data Distribution System

Rainer Ch. Koppler, Gerhard Kurka, and Jens J. Volkert

Department of Computer Graphics and Parallel Processing
Johannes Kepler University, A-4040 Linz, Austria/Europe

Abstract. This paper introduces *Exdasy*, a user-friendly and extendable software tool for partitioning unstructured meshes and mapping mesh partitions to parallel computers. Exdasy was designed to meet the increasing demands to today's data distribution systems, which are posed by the variety of mesh computations, the ongoing development of distribution algorithms and rapid changes in parallel hardware technology. For this, Exdasy offers third-party state-of-the-art distribution algorithms augmented with graphical user interfaces and powerful graphical evaluation displays. Evaluation of distributions is based on various quality metrics and static machine parameters. Exdasy provides a modular architecture by means of replaceable distribution algorithms, machine models and evaluation facilities. Hereby it is attractive to both users and developers.

1 Introduction

In scientific computing unstructured meshes are frequently used, for example, in computational fluid dynamics or structural analysis applications. Nowadays computations with large unstructured meshes are performed on parallel computers because of size considerations and computational requirements. With distributed memory machines the user is faced with the non-trivial problem of data distribution. Meshes must be partitioned across the local memories of the processors in such a way that a good load balancing is achieved and interprocessor communication is kept minimal. It is known that optimization of these issues involves the solution of NP-complete problems [5], so up to now researchers have developed heuristics that yield reasonable, yet suboptimal results [8] [18] [9] [15]. With virtual shared memory machines data distribution is also relevant since a good distribution promotes data locality and so improves overall performance.

The decision about the *efficiency* of a distribution is hard to find because it depends on several factors, including the suitability of a distribution algorithm for the given mesh, the problem to be solved, and the parallel computer. Thus the data distribution procedure is not a fully automatic, but an interactive process, which should be supported by appropriate tools. With such a tool application users can create and evaluate several distributions and determine the best one for their purposes.

In this paper we introduce *Exdasy* (Extendable Data Distribution System), a user-friendly and extendable software tool for partitioning unstructured meshes and mapping mesh partitions to parallel computers. Exdasy offers state-of-the-art distribution algorithms, which were developed by other researchers, augmented with graphical user interfaces. It provides exploration and evaluation of distributions by means of several 2D and 3D color graphics displays. Evaluation of distributions is based on distribution quality metrics and static target machine parameters. Unlike existing systems, Exdasy provides both evaluation of single distributions and comparison of several distributions with respect to various metrics. It is dedicated to application users and distribution algorithm developers. Its highly modular design allows easy adaptions with respect to distribution algorithms, target computers, mesh input/output formats, and graphical displays. Finally, Exdasy's implementation promotes portability across a wide range of operating systems and graphics hardware.

In the following, we discuss requirements to today's data distribution systems and show their realizations within Exdasy. We describe the graphical facilities of Exdasy and outline their use for the data distribution procedure. Furthermore we show Exdasy's extendability and first experiences by means of short examples. A comparison of Exdasy with related systems and a brief view on future activities concludes this paper.

2 Requirements to Data Distribution Systems

The data distribution problem is usually divided into two subproblems, which are both NP-complete: *partitioning* and *mapping*. During partitioning the data domain is divided into several parts such that all parts have equal sizes and the boundaries between adjacent parts show minimal lengths. The result is called a partition, the respective graph is called the partition graph. During mapping the partition is mapped onto the target computer such that adjacent parts are assigned to adjacent processors in the interconnection network. This two-step approach separates machine-independent issues of the data distribution problem from machine-dependent issues. Consideration of the mapping step turned out to be important with medium-size and massively-parallel systems [19]. Today many implementations subsume both partitioning and mapping [8] [7]. In the following the term *distribution* stands for a partition that has been mapped onto a target computer.

With an *efficient* distribution a given parallel computation yields good performance with a given input mesh on a given target computer. Objective functions of distribution algorithms cannot precisely consider each of these factors because of the variety of target machine and application characteristics. Thus users need a tool that allows easy experiments with different partitioning and mapping algorithms. The tool should provide easy exploration and accurate evaluation of distributions such that users quickly find a suitable algorithm. For this, it should support both investigation of single distributions and comparison of several distributions, preferably through graphical means.

Both partitioning and distribution evaluation should consider application characteristics in order to yield usable results. For example, users may want to partition either the nodes or the elements of a mesh. On the other hand, evaluation should be based on metrics that relate to application characteristics and consider target machine properties. Thus both the application and the target computer must be modelled with a sufficient degree of detail.

Besides these issues, user acceptance is promoted if a tool can process various mesh formats and write results in several output formats. Such a facility allows easy integration of mesh distributions in applications and thus simplifies distribution evaluation through experimental runs.

Occasionally some facilities of a tool are not satisfactory, for example, obsolete distribution algorithms shall be replaced by better ones, or existing evaluation or mesh input/output facilities are inappropriate for the given purpose. Thus the tool should exhibit a modular rather than a monolithic structure such that single components can be replaced easily, preferably without re-building the whole system. For this, developers require a comfortable application programming interface (API), which covers distribution algorithms, mesh formats, target computer models and evaluation displays. Each new component that a developer integrates into the tool can benefit from existing facilities. For example, graphical evaluation displays may assist a developer with detecting errors or weaknesses in his or her distribution algorithm. Another issue, which is derived from this example, is robustness. If a tool is robust against crashes caused by one of its components, for example, if distribution algorithms are executed within separate processes, users are more willing to work with it.

3 Overview of Exdasy

Exdasy is a general-purpose data distribution system for unstructured mesh applications. The created distributions are collected in workspaces and can be evaluated with respect to their performance within a given parallel application. By means of state-of-the-art distribution algorithms and powerful graphical evaluation facilities the system offers users a quick and easy way to finding appropriate distribution algorithms.

Development of Exdasy was primarily motivated by the deficiencies of existing data distribution systems, including *DecTool* [2], *TOP/DOMDEC* [3], and *DDT* [4]. The ongoing development of partitioning and mapping algorithms and rapid changes in parallel hardware technology pose new requirements to data distribution systems. Exdasy considers these requirements already in its basic design since it can be configured with different distribution algorithms, target computers, mesh formats and evaluation displays. So it provides a powerful infrastructure for future developments.

During a session with Exdasy the user loads a mesh and creates a number of partitions using the available packages, among them the popular partitioners *Chaco* [8] and *Metis* [9]. Optionally, partitions can be mapped onto a selected target computer by means of an available mapper, for example, the CPE heuristic [6]. All mapped and unmapped partitions can be explored and evaluated through graphical displays, which are chosen from a menu. Evaluation is based on the selected target computer, which can be changed through a pop-up menu. Each distribution in the workspace can either be saved or further optimized by another partitioning or mapping algorithm.

Exdasy lets the user specify application characteristics for partitioning by means of *vertex weighting*, which is applied to the graph derived from the loaded mesh. The graph vertices are abstractions of mesh vertices or finite elements, which depends on the given mesh format and user settings. Exdasy accepts meshes given in NASTRAN's format [13], *OFF* (Object File Format) [16], Chaco's format, and can be extended with further modules.

Partitioning and mapping algorithms in Exdasy provide intuitive graphical front-ends with online help and are chosen from separate menus. Exdasy executes algorithms in separate child processes in order to ensure robustness of the whole system and to prevent waiting times.

4 Graphical Facilities

Exdasy provides exploration and evaluation of mesh distributions through several kinds of viewers, which are classified into *graph viewers* and *chart viewers*. The viewers use colored 3D graphics in order to visualize quality metrics that are related to partitions, mappings and target computers. Exdasy's built-in metrics consider different computation and communication characteristics of common scientific applications. More specific metrics can be incorporated by means of additional chart viewers. Some important built-in subgraph-related and connection-related metrics are given in the sequel. If weighted graphs are considered, node and edge *numbers* must be thought of as *weights*.

1. *internal node number* (subgraph): load balancing metric for applications with nodewise computations, for example molecular dynamics codes [1].
2. *internal edge number* (subgraph): load balancing metric for applications with edgewise computations, for example Euler solvers [11] or matrix-vector multiplication.
3. *border node number* (subgraph): communication metric for nodewise computations.

4. *cut edge number* (subgraph): communication metric for edgewise computations.
5. *neighbour number* (subgraph): important communication metric for target computers with long message startup times.
6. *communication cost* (subgraph): sum of the cut edge number with each adjacent subgraph weighted by the distance between the subgraphs in the target computer.
7. *single-hop neighbourship ratio* and *single-hop communication cost ratio* (subgraph): important communcation metrics for target computers where global communication is expensive compared to nearest-neighbour communication.
8. *cut edge number*, *hop number*, and *cut edge number weighted by hop number* (connection).

Each viewer shows metric values as spheres or bars. Both the size and the color of such an object depict values, either values of the same metric or of two different metrics. Object with low diameters or with "cool" colors (such as *blue)* show low metric values, whereas high diameters and "hot" colors (such as *red*) are used for showing high metric values. We have chosen such a coloring scheme since optimization, as it is concerned with data distribution, usually aims at minimization. So bottlenecks are always highlighted such that users can locate them quickly.

4.1 Graph Viewers

As previouly mentioned, Exdasy derives a graph representation from input meshes. Graph viewers associate a graph representation with various *properties*, which primarily include distribution quality metrics. Such viewers draw graph vertices and graph edges with different colors and in different sizes and thus show the distribution of property values across the entire graph in one picture. Hereby the user can quickly locate bottlenecks.

All graph viewers of Exdasy have a uniform appearance and provide common facilities since they are based on a generic graph viewer implementation. The generic facilities include drawing of arbitary graphs, basic view manipulation such as translation, zooming and rotation, slicing, and scene configuration according to *OpenGL*'s [14] facilities.

Figure 1 shows a *distribution viewer*, an important and frequently used graph viewer, which provides detailed exploration and evaluation of of distributions. It draws a distribution either as an explosion representation of the mesh geometry or as a partition graph. The *explosion representation* augments the mesh geometry with colored connections between subgraphs. The explosion degree and connection diameters can be changed. Optionally only vertices along subgraph boundaries are shown. The subgraph coloring is either arbitrary or can be associated with one of the built-in metrics. For connections between subgraphs a similar facility is offered, with the difference that diameters and colors may be associated with different metrics. Furthermore connections can be hidden completely or drawn dotted, where the dot number corresponds to the hop number between the subgraphs in the given target computer. In Fig. 1 both subgraphs and connections are colored according to their communication costs. Thus subgraphs in the front and in the back of the aeroplane have cool colors, whereas subgraphs along the wings are drawn with hot colors in order to warn the user of large message sizes.

The *partition graph representation*, as shown in Fig. 2a, collapses subgraphs to spheres. This representation is particularly useful when the user is primarily interested in connections. For example, in Fig. 2a high communication costs between subgraphs along the wings can be detected easier than in Fig. 1 (with neglection of the subgraph coloring).

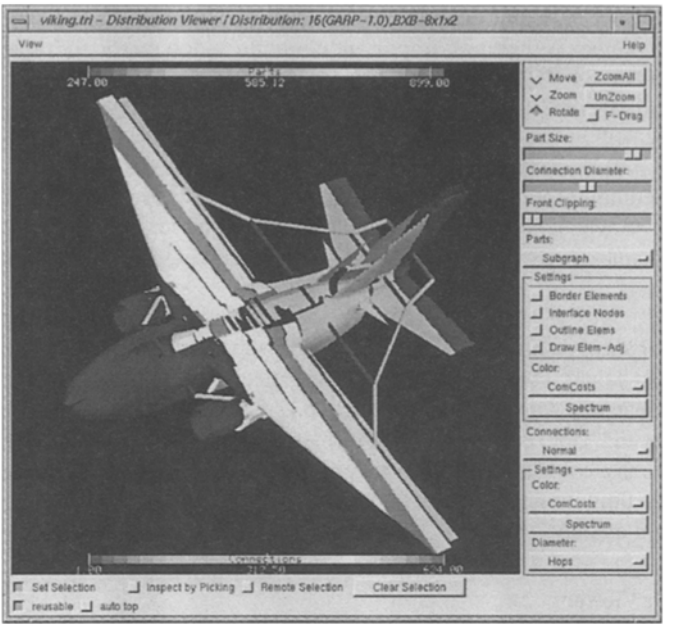

Fig 1: distribution viewer (explosion representation)

The graph viewer also provides detailed examination of subgraphs and connections. When the user selects such an object and the *Inspect by Selection* option is set, Exdasy opens a window that shows details about the selected object. Subgraph windows, as shown in Fig. 2b, also show the subgraph geometry, which can be moved, zoomed, or rotated separately.

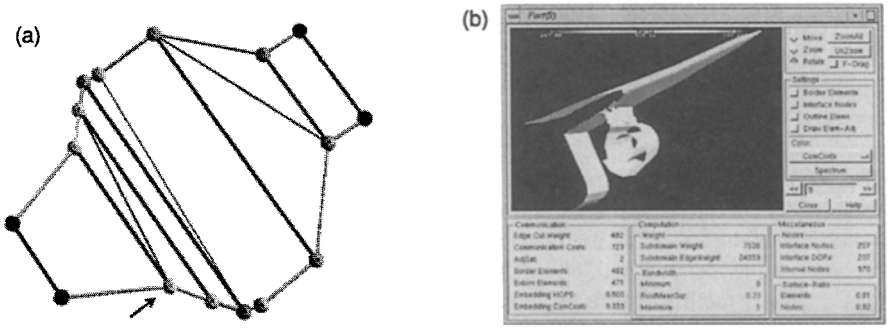

Fig 2: (a) distribution viewer (partition graph representation), (b) examination of a subgraph

The *grid viewer* and the *data viewer* are further useful graph viewers. The grid viewer allows exploration of a mesh without a given partition. It is typically used after loading of a mesh in order to select appropriate partitioning algorithms. For example, it can draw disconnected parts in different colors, which tells the user not to use graph-oriented partitioning algorithms. The data viewer colors mesh nodes or elements according to data values that can be read from files. It is typically used for visualization of solution values. Whenever scalar data shall be displayed, the user does not need to switch to another environment.

4.2 Chart Viewers

Chart viewers visualize distribution quality metrics by means of charts. They are helpful when information displayed in graph viewers is hard to survey or the impact of bottlenecks located in the graph viewer on program performance shall be evaluated. Thus chart viewers reduce large amounts of data to meaningful graphical abstractions and so allow quick evaluation, even with large meshes and massively parallel computers. For this, they provide the same basic view manipulation facilities as graph viewers.

Fig. 3 shows two instances of the *matrix viewer*, a 3D chart that shows the communication pattern for a distribution when the parallel application performs a local pairwise exchange operation. For each pair of communicating processors a bar is drawn, where the height depicts either the cut edge number, the hop number or the cut edge number weighted by the hop number. In Fig. 3 the last metric is chosen. Bar colors always depict distances between processors. The user can select single bars in order to determine metric values. For example, in Fig. 3 the user selected the highest bars in two distributions of the mesh shown in Fig. 1, and got the given values. The left distribution's bottleneck, which characterizes the long boundary between subgraphs along wings (see Fig. 1), is much greater.

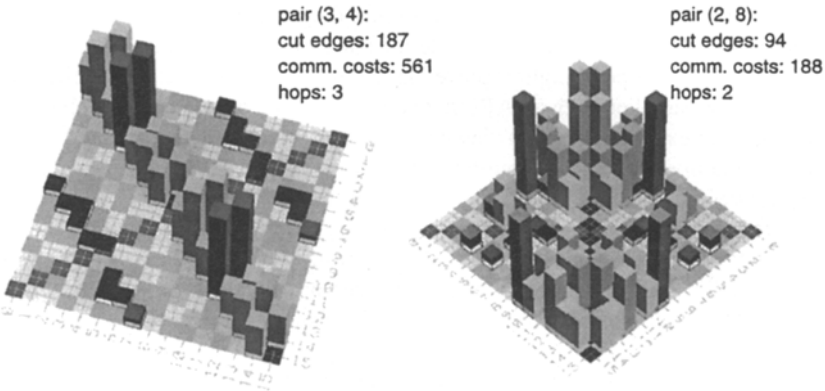

Fig 3: matrix viewers: (a) simple 8 x 2 distribution, (b) Metis distribution for 16 processors

The *metric viewer* shows metrics of the currently investigated distribution in table form. It reports minimum, average and maximum values for each metric listed at the beginning of section 4. Furthermore it extends subgraph metrics to the entire graph. The metric viewer is a useful facility when distributions shall be compared by means of numbers or deviations of bottlenecks from averages shall be quantified.

The *comparative metric viewer* is a 3D chart that shows selected metrics of *all* distributions given in the workspace. The first dimension enumerates the selected metrics, the second dimension ranges over all distributions, and the third dimension shows metric values by means of colored bars. The height and the color of a bar is chosen with respect to the maximum value of the respective metric. Thus the worst distribution is associated with the "hottest" and highest bar. While the matrix viewer and the metric viewer apply to single distributions, the comparative metric viewer can evaluate the entire workspace and compare several distributions to each other. It assists the user with making a quick decision between a number of alternatives with respect to metrics that have proven useful concerning the given application. It is also useful for experiments, for example, in cases where the user wants to evaluate the impact of different distribution algorithm settings.

Besides that the user can compare sets of distributions by means of multiple instances of other viewers. For this, each viewer kind provides a *reusable* option. Whenever a viewer has this option set, it can repeatedly be used for showing different objects, i. e., graphs or distributions. If the option is not set, the viewer is tied to its current object, and the system opens an additional viewer for the visualization of other objects. This facility provides detailed comparison of several distributions, in contrast to the comparative metric viewer, which simplifies global comparison.

5 Configuration and Extensions

As mentioned in section 2, today's data distribution systems should provide enough flexibility such that users can configure them for particular purposes and extend them with facilities that are not (or *should* not) be provided by the system itself. For this, Exdasy offers several *internal* interfaces and an *external* interface to other systems.

The internal interfaces allow replacement of various components, including partitioners, mappers, mesh readers, target computer models and evaluation displays. Each component is implemented as a shared library, which is located in a special directory and linked to Exdasy during startup. Users will typically configure the system with their own mesh formats and a model of their target computer while utilizing existing algorithms and displays. By way of contrast, developers might replace algorithms or add application-specific displays. For this, they are supplied with an API that consists of C++ base classes for all replaceable components. For example, the grid viewer and data viewer were implemented in this way.

The external interface allows coupling of Exdasy with other systems. For example, the existing viewer set, which provides exploration at the mesh and application levels, can be extended to the network level of the target computer. Hereby the user gets a realistic picture of message hops and network conflicts. *MuCH* (Multiprocessor Class Hierarchy) [10], a tool for modelling and visualization of multiprocessors, is ideally suited for this purpose. Besides facilities that realize a *network viewer* for Exdasy it offers also an external interface.

Figure 4 shows a seamless cooperation between Exdasy and MuCH. The user evaluates a mesh distribution onto a nCUBE 2 with 16 processors, where network conflicts are of primary interest. When the *Remote Selection* option in the distribution viewer is chosen, Exdasy propagates selections of subgraphs and connections to MuCH, which marks processors and communication paths, respectively. In Fig. 4 the user successively selects the following subgraph connections, which cause more than one message hop: (12, 15), (4, 15) and (6, 15). The selections can be performed in the matrix viewer as well since each selection causes updates in all open viewers. The first two communication paths are established without conflicts, the third one gets blocked on processor 7. Accordingly MuCH draws the first two paths in green, the third one in red and only up to processor 7.

Systems that connect to Exdasy's external interface must adhere to our communication protocol named *GDI* (Generic Data Interchange). GDI is similar to *ToolTalk* [17] in that participants do not communicate directly but via a server, regardless of their physical locations.

6 Experiences

In the following we report about first experiences with Exdasy. We have collected them with a parallel Euler solver [11], which was implemented with MPI [12] and runs on the

nCUBE 2. The kernel of this application is a loop over all mesh edges, where each iteration computes flux contributions for both incident vertices.

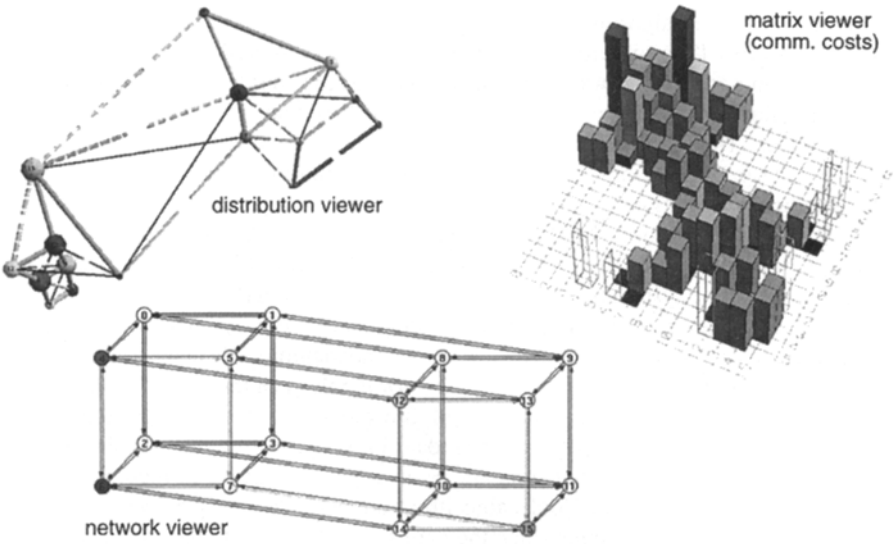

Fig 4: Interoperating viewers of *Exdasy* and *MuCH*

At first we determined application characteristics of the solver and utilized our knowledge about the nCUBE 2 in order to find metrics that describe the efficiency of a data distribution for the Euler solver when it runs on the nCUBE 2. Two metrics, *maximum cut edge number of a subgraph* and *average subgraph degree* (i. e., number of adjacent subgraphs), showed reasonable accuracy.

After a few experiments with 64 processors, where we used a mesh with 62K vertices and 366K edges, we learned that the partitioner Metis is the best choice for this application. Furthermore we used the CPE heuristic to map the partition onto a hypercube. Surprisingly it showed that the optimized mapping causes slight performance degradation, whereas the communication cost metric predicted a gain of 13 percent. The combination of Exdasy's matrix viewer with MuCH, as shown in the previous section, gave us an insight into this phenomenon. In the matrix viewer we arbitrarily selected some of the highest bars in order to determine network conflicts through MuCH's network viewer. With the unmapped partition all communication paths could be established without conflicts, whereas with the CPE mapping conflicts arose after few bars had been selected.

Summing up, with respect to the given simple application the viewers of Exdasy and MuCH allow quick and easy identification of performance bottlenecks a priori, and thus are more useful than conventional distribution metrics.

Our experiences with Exdasy also cover technical issues. Mesh I/O performance, visualization performance and memory management strongly affect the usability of a data distribution system. Several redesigns and optimizations were applied to Exdasy in order to make it applicable for large meshes and to achieve reasonable performance even on low-end workstations. For example, on a Linux PC with 32 MB memory Exdasy shows good "interactivity" with the mesh shown in Fig. 1 (17K vertices and 55K edges).

7 Related Work

Up to the present only few data distribution systems have been developed that integrate a number of distribution algorithms with graphical evaluation facilities.

TOP/DOMDEC [3] is regarded as a state-of-the-art mesh partitioning environment. It offers algorithms for initial partitioning of graphs and for non-deterministic optimization of graph partitions. Optimization of partitions can be based on various single or combined quality metrics. The system does not support separate mapping of partitions to machine topologies. It shows partitions similar to Exdasy's distribution viewer and provides partition evaluation by means of various metrics. These cover also computations on data structures that are derived from the mesh representation, for example, stiffness matrices. From today's point of view, most algorithms provided by TOP/DOMDEC are obsolete. This is not a drawback of the system itself since algorithms can be replaced. Although based upon IRIS GL, the evaluation displays are of quite poor quality. For example, the *interconnectivity map* (similar to Exdasy's matrix viewer) is a monochrome 2D display. Finally, evaluation of partitions does not consider the processor topology of the given target computer.

DecTool [2] is a small data distribution tool. It offers several built-in algorithms for partitioning of graphs and mapping of partitions to arbitrary machine topologies. Partitions can be shown in two and three dimensions and even altered by means of mouse operations. From today's point of view, DecTool's algorithms are partially obsolete. DecTool does not support replacement of algorithms since they are statically linked to the tool. The partition display is quite simple since it is implemented upon X11 directly. View manipulation is not supported, evaluation facilities are missing.

DDT (Domain Decomposition Tool) [4] is a very simple data distribution tool. It has six built-in algorithms that are not comparable with today's distribution algorithms. The tool also lacks arbitrary view manipulation and provides very limited evaluation facilities.

8 Conclusions

We presented Exdasy, a user-friendly and extendable software tool for partitioning unstructured meshes and mapping mesh partitions to parallel computers. Development of such a tool is motivated primarily by the ongoing development of parallel unstructured mesh applications and rapid changes in parallel hardware technology, which pose requirements that are not satisfied by today's data distribution systems. Users are confronted with the problem of choosing algorithms that distribute their meshes across the processors of a given parallel computer such that their applications achieve optimal performance. Exdasy considers this problem in its basic design and offers a powerful infrastructure for environments where users can easily find good distributions by means of graphical means and accurate models, and where developers can integrate and test their own developments.

Our future activities will focus on development and integration of additional components, primarily distribution algorithms (for example, *Jostle* [18] and *Scotch* [15]) and application-specific evaluation facilities. Unlike TOP/DOMDEC, Exdasy does not provide evaluation of distributions for *implicit* computations [3], i. e., computations that are not concerned with mesh entities but with a derived matrix. In order to allow accurate evaluation with respect to such applications Exdasy must provide additional metrics and graphical displays. For this, we plan close cooperations with application code developers.

References

1. Brooks B. et al.: "CHARMM: A Program for Macromolecular Energy, Minimization and Dynamics Calculations", Computational Chemistry, Vol. 4, pp. 187-217, 1983.
2. Chrisochoides N., Houstis C., Papachiou P., Houstis E., Kortesis S., Rice J.: Domain Decomposer: A software tool for mapping PDE computations to parallel architectures", in Glowinski R. et al. (Eds.): Domain Decomposition Methods for Differential Equations IV, pp. 341-357, SIAM, Philadelphia, 1991.
3. Farhat C., Simon H. D.: "TOP/DOMDEC – a Software Tool for Mesh Partitioning and Parallel Processing", Technical Report RNR-93-011, NASA Ames Research Center, Moffett Field, CA, 1993.
4. Floros N., Reeve J.: "Domain Decomposition Tool – An abridged User's Guide", Department of Electronics and Computer Science, University of Southampton, 1994.
5. Garey M. R., Johnson D. S.: *Computers and Intractability: A Guide to the Theory of NP Completeness*, Freeman, New York, 1979.
6. Hammond S.: *Mapping Unstructured Grid Computations to Massively Parallel Computers*, Ph.D. thesis, Department of Computer Science, Renesselaer Polytechnic Institute, 1992.
7. Heiss H.-U., Dormanns M.: "Partitioning and mapping of parallel programs by self-organization", Concurrency – Practice and Experience, Vol. 8, pp. 685-707, 1996.
8. Hendrickson B., Leland R.: "The Chaco User's Guide Version 2.0", Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque, NM, 1995.
9. Karypis G., Kumar V.: "METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System - Version 2.0", Department of Computer Science, University of Minnesota, 1995.
10. Kranzlmüller D., Koppler R., Grabner S., Holzner Ch., Volkert J.: "Parallel Program Visualization with MUCH", in L. Boeszoermenyi (Ed.):, ACPC '96 - Parallel Computation, Proc. of the 3rd International ACPC Conference, pp. 148-160, LNCS 1127, Springer Verlag, 1996.
11. Mavriplis D.J.: "Three-Dimensional Multigrid for the Euler Equations", AIAA Paper 91-1549CP, American Institute of Aeronautics and Astronautics, Washington D.C., pp. 824-831, 1991.
12. M.P.I. Forum: "MPI – A Message Passing Interface Standard", Computer Science Technical Report CS-94-230, University of Tennessee, 1994.
13. The MacNeal-Schwendler Corporation: *MSC/NASTRAN User's Manual*, Vol. 1, Los Angeles, CA, 1991.
14. OpenGL Architecture Review Board: *OpenGL Reference Manual*, Addison-Wesley, 1992.
15. Pellegrini F., Roman J.: "SCOTCH: A Software Package for Static Mapping by Dual Recursive Bipartitioning of Process and Architecture Graphs", in Liddel H., Colbrook A., Hertzberger B., Sloot P. (Eds.): Proc. High-Performance Computing and Networking '96, pp. 493-498, LNCS 1067, Springer Verlag, 1996.
16. Phillips M.: "GeomView User Manual", available through http://www.engr.usask.ca/macphed/finite/fe_resources/node121.html, 1994.
17. SunSoft Inc.: *The ToolTalk Service: An Inter-Operability Solution*, SunSoft Press, 1996.
18. Walshaw C., Cross M., Johnson S., Everett M.: "JOSTLE: Partitioning of Unstructured Meshes for Massively Parallel Machines", in Proc. Parallel CFD'94, Kyoto, 1994.
19. Walshaw C., Cross M., Everett M., Johnson S., McManus K.: "Partitioning and Mapping of Unstructured Meshes to Parallel Machine Topologies", in Ferreira A., Rolim J. (Eds.): Parallel Algorithms for Irregular Structures Problems, Proc. Irregular '95, pp. 121-126, LNCS 980, 1995.