Partition Alignment in Three Dimensional Unstructured Mesh Multi-Physics Modelling

Kevin McManus, Chris Walshaw, Steve Johnson, Mark Cross

k.mcmanus@gre.ac.uk
Centre for Numerical Modelling and Process Analysis
University of Greenwich, London, SE18 6PF, UK

Unstructured mesh codes for modelling continuum physics phenomena have evolved to provide the facility to model complex interacting systems. Parallelisation of such codes using single Program Multi Data (SPMD) domain decomposition techniques implemented with message passing has been demonstrated to provide high parallel efficiency, scalability to large numbers of processors P and portability across a wide range of parallel platforms. High efficiency, especially for large P requires that load balance is achieved in each parallel loop. For a code in which loops span a variety of mesh entity types, for example, elements, faces and vertices, some compromise is required between load balance for each entity type and the quantity of inter-processor communication required to satisfy data dependence between processors.

1. Introduction

Work at the University of Greenwich has for some time been focussed on the modelling of multiple interacting physical processes in a range of applications, for example, metals casting [2,3,1]. Prediction of the quality of a metal casting necessitates accurate modelling of all of the physical phenomena. These include free surface flows, thermal circulation currents, heat conduction, latent heat of solidification, solid deformation and stress, gap formation, heat transfer across gaps, contact analysis and other phenomena. The complexity of the problem geometry has led to the development of Finite Volume Unstructured Mesh (FV-UM) techniques for the solution of the Navier Stokes equations with close coupling to a FV-UM solution of the elastic visco-plastic stress strain equations. Extension of the methods into three dimensions has been enabled through the development of PHYSICA, a framework for supporting three dimensional, unstructured mesh, continuum mechanics modelling (http://physica.gre.ac.uk).

The difficulties encountered in implementing large scale CM codes such as PHYSICA on multiprocessor systems are now fairly well understood. Despite the claims of shared memory architecture manufacturers to provide effective parallelising compilers, these have not proved to be adequate for large or complex programs. The paradigm of Single Program Multi Data (SPMD) domain decomposition with message passing, where each processor runs the same code on a subdomain of the problem, communicating through exchange of messages, has for some time been demonstrated to provide the required level of efficiency,

scalability and portability across both shared and distributed memory systems, without the need to re-author the code into a new language [4].

2. Load Balance and Communication

Parallel PHYSICA uses SPMD techniques to provide potentially high performance on a wide variety of parallel platforms. The domain decomposition method employed is to partition the computational mesh into P parts which are distributed on to P processors. Data that is required in one partition (processor) but is calculated in another partition creates a requirement for communication between the processors. This is accommodated through the use of mesh overlaps.

A number of parameters affect parallel performance. Of great importance is the data distribution defined by a mesh partition. Much effort has been devoted to the development of mesh partitioning techniques that return a high quality partition for low cost [7]. Definition of partition quality has itself been a subject for much discussion [5]. The reduction of interprocessor communication through reduction of the partition surface while maintaining load balance has been of primary interest. More recently the development of localised conjugate gradient preconditioners has led to interest in the aspect ratio of the partition [6]. Inter-processor edge weight, balance and aspect ratio are however only predictors of partition quality. The best partition is that which returns the shortest run time. Correlation between predictors and actual performance can become extremely complex.

Computational load is seldom homogeneous and so weighting systems are employed to improve predictors. Inter-processor communications are inevitably imbalanced and often significantly non-deterministic. For a three dimensional unstructured mesh code such as PHYSICA the situation is further complicated by calculations being performed on variables that are associated with one of three mesh entity types; vertices, faces and elements. Parallel PHYSICA consequently requires three partitions, one for each mesh entity type. This introduces the issue of alignment between each partition. Each individual mesh entity exists exclusively in one partition (some codes use shared entities but complex interactions preclude use of this technique for PHYSICA). Ideally each vertex that defines a face will exist in the same subdomain as the face. Similarly each face of an element will exist in the same subdomain as the element. The topology of an unstructured mesh means that this is almost impossible to achieve. At the surface of each subdomain there will exist, for example, faces that contain vertices and elements that consist of faces, both from neighbouring subdomains. If load balance is to be maintained for each mesh entity type then this mis-alignment can become severe and may lead to a subdomain containing, for example, a face for which all of the vertices are in neighbouring subdomains. Such mis-alignment infers a communication requirement. For example calculation of the enthalpy for an element requires knowledge of the heat flux through each of the element's faces. If an element's face is not in the same subdomain as the element then it's flux, which is calculated by the processor that owns the face, must be communicated from that processor. These conditions result in a compromise being forced between load balance and communication.

3. Partitioning

PHYSICA calls the graph partitioning code JOSTLE [7] to provide mesh partitions at run time. JOSTLE uses a number of techniques (graph theoretic, heuristic, multi-level) to rapidly provide a near optimal graph partition. Data dependencies in the unstructured mesh are are passed to JOSTLE as a graph $G(N, E)$ of nodes N and edges E where each node represents a mesh entity and each edge represents an entity to entity dependence. The mesh partition quality is dependent on how the mesh is represented as a graph to JOSTLE. For example, the workload associated with each element may be proportional to the number of vertices in the element. A graph can then be created in which each vertex represents a mesh element, each edge represents element connectivity and each vertex in the graph is given a weight according to the number of vertices in the element it represents. The partition returned from JOSTLE will then be a useful partition for the mesh elements, this is referred to as a primary partition. Partitions for the other mesh entity types, secondary partitions, can then be derived from the primary partition. So, for example, with a primary partition based on elements a secondary partition for vertices may be derived by assigning each vertex to the subdomain that contains the greatest number of elements containing the vertex. This method provides a good alignment between the partitions but can result in a poor load balance in the secondary partitions especially if the element degree varies across the mesh. Clearly the mesh entity type associated with the greatest amount of work should be chosen to provide the primary partition so that a load imbalance in the secondary partitions is less significant than any load imbalance in the primary partition.

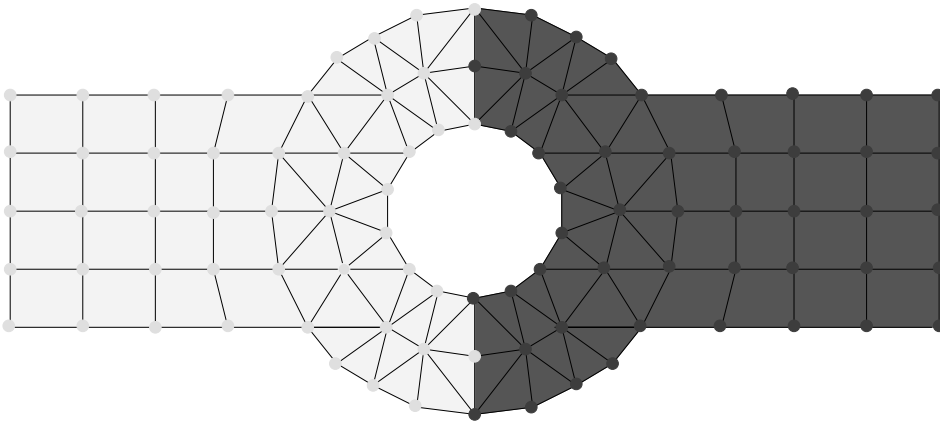

Figure 1. Mesh of 102 elements and 94 vertices in two partitions each with 51 elements and 47 vertices.

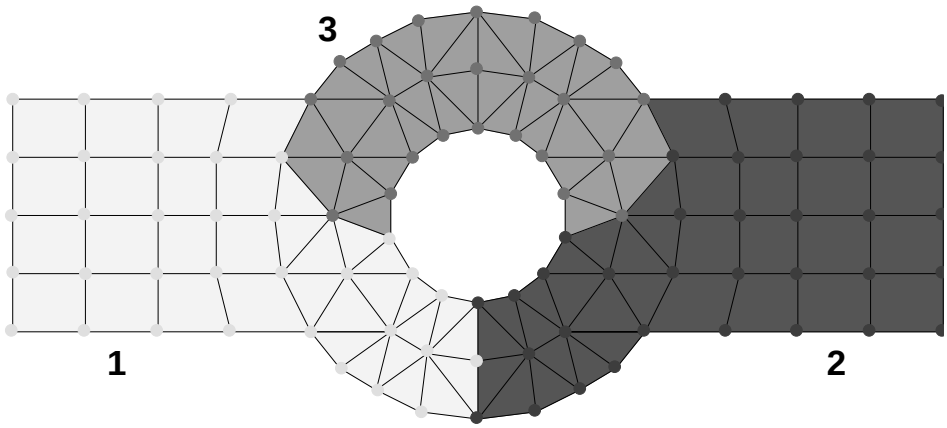

Figure 2. Mesh of 102 elements and 94 vertices in three partitions each with 34 elements and 34, 35 and 25 vertices.

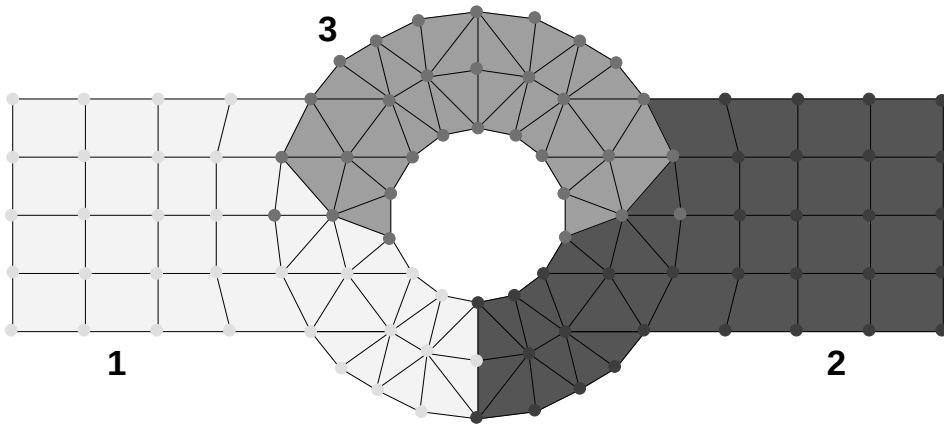

Figure 3. Mesh of 102 elements and 94 vertices in three partitions each with 34 elements and 31, 32 and 31 vertices.

4. Partition Alignment

Consider the unstructured (2D) mesh of 102 elements and 94 vertices shown in Figure 1. With a primary partition based on elements and a secondary partition for vertices then for two subdomains the number of elements and vertices is balanced in each subdomain. With the mesh split into three subdomains as in Figure 2 the number of vertices in each subdomain is imbalanced. This imbalance can be corrected by moving vertices from subdomains 1 and 2 into subdomain 3 on the basis of vertex to element connectivity and vertex to vertex connectivity. The secondary partition resulting from such rebalancing is shown in Figure 3. Now one element from subdomain 1 and one element from subdomain 2 each consist of vertices belonging entirely to subdomain 3. The effect of this partition misalignment on the required overlap and hence data communication is illustrated in Figure 4.

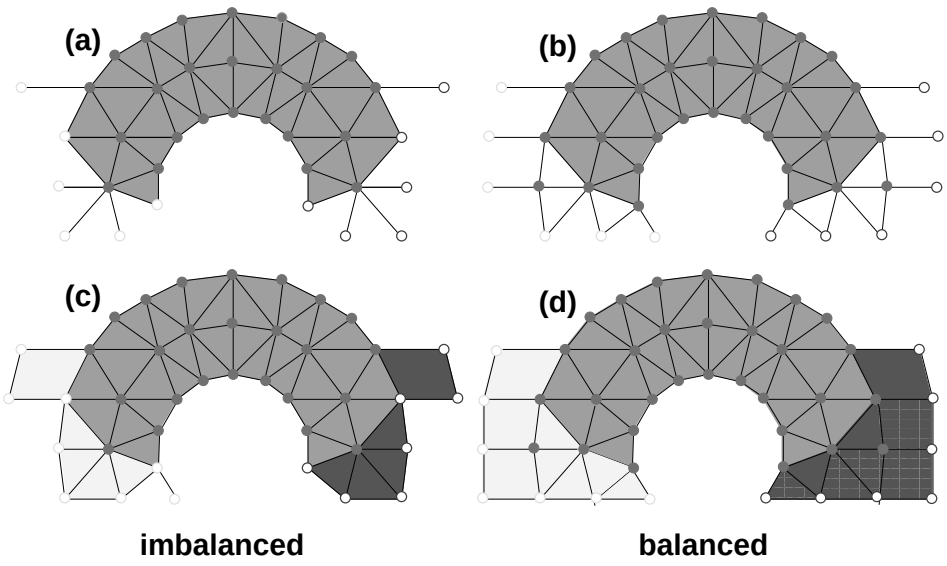

Figure 4. Vertex and element overlaps for subdomain 3.

For an application in which the data dependence requires a vertex overlap defined as: each vertex connected to an owned vertex, then the vertex overlaps will be as shown in Figure 4 (a) and (b). Here the number of vertices required in the overlaps for the imbalanced and balanced secondary partitions are the same and so load balance has been achieved without incurring any communication overhead. In a more realistic case, with a data dependence such as in PHYSICA (although simplified for this example), where there are also element based data dependencies to consider:

- A vertex is required in the overlap if:-
  The vertex is connected to an owned vertex.
  The vertex is required by an owned or an overlap element.

- An element is required in the overlap if:-
  The element is adjacent to an owned element.
  The element requires a least one owned vertex.

These dependencies lead to the overlaps illustrated in Figure 4 (c) and (d). Now a greater number of both vertices and elements is required in the overlaps for the balanced secondary partition. In this case the amount of data to be communicated in an overlap update has increased but a balance has been achieved for both the primary and secondary partitions. The effect of the increase in communicated data is dependent on many factors:

- The ratio of the inter-processor communication bandwidth to the calculation speed of the parallel machine.
- The degree of load imbalance in the secondary partition(s), being largely a consequence of the nature of the unstructured mesh.
- Extending the depth of subdomain overlap can result in dependencies between previously unconnected subdomains and so incur an additional inter-processor communication start-up latency.

Given the highly platform, mesh and application dependent effect of partition alignment and secondary load balancing it is optimistic to expect performance predictors to be accurate. Inhomogeneous computational load for each mesh entity further complicates these problems.

5. Results and Conclusions

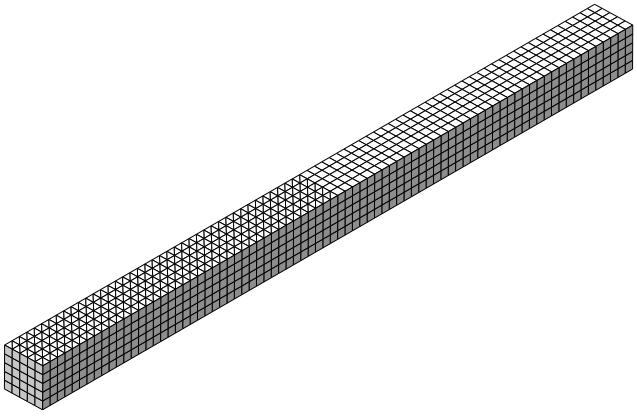

Figure 5. Mesh of mixed element types.

Test cases representative of multi-physical applications have shown the potential for improved parallel performance. In all cases there is a compromise between the idle time

incurred due to a load imbalance in secondary partitions and the time required to communicate the additional overlap data resulting from the increased overlap depth. For many applications, performance is limited primarily as a result of load imbalance and communication start-up latency and so it is usually worthwhile to trade load imbalance for communication time. However, there is also the potential for a degradation in performance, for example, applications that are bandwidth limited such as large test cases on workstation networks with poor interconnection.

The mesh in Figure 5 was constructed to demonstrate the problem of partition alignment. This mesh is run with a multi-physics problem having a mix of element, vertex and face based loops. A similar geometry contains 44,775 vertices, 162,358 faces and 58,212 elements, of which 19,404 are hexahedral and 38,808 are pentahedral. Using JOSTLE to calculate a primary partition for elements ensures an excellent balance in element numbers. So for two processors each processor is allocated 29,106 elements but this means that with aligned secondary partitions each processor is allocated 27,844 and 16,931 vertices, an imbalance of approximately 3:2. This imbalance leads to a poor speed-up even for small numbers of processors. With eight processors this situation has reached the point at which the vertex partition imbalance has become almost 2:1 with each subdomain having between 8325 and 4401 vertices. The effect of this is reflected in the speed-up curve in Figure 6 where the best possible speedup for vertex based loops cannot exceed 50% efficiency.

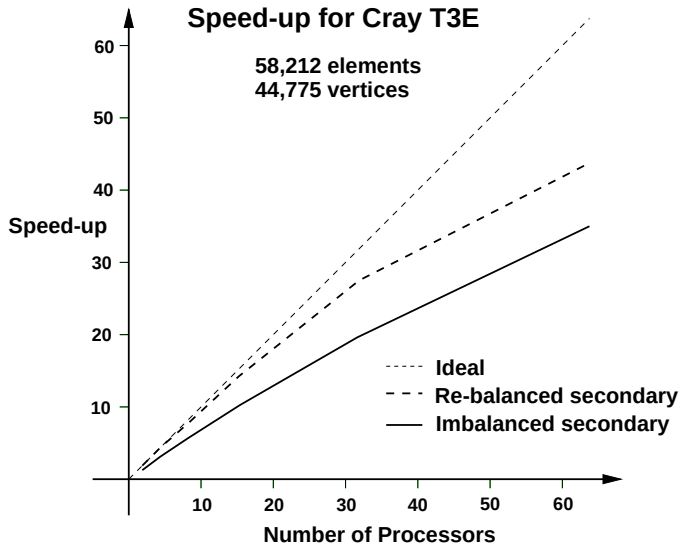

Figure 6. Speedup for imbalanced and balanced secondary partitions on a Cray T3E.

Figure 6 shows that imbalance in secondary partitions limits performance even for small numbers of processors but good scalability (near constant gradient) is possible as the imbalance has reached a limit, in this case, at around eight processors, after which the imbalance for vertices remains approximately 2:1. The speed-up improvement for

re-balanced partitions is good but the gradient of the curve suggests that this will not continue to scale. This is not surprising as the increasing misalignment between the partitions leads eventually to some subdomains having no alignment between the partitions for each entity. That is a subdomain in which no core element contains a core vertex. This results in overlaps that are significantly larger than the subdomain core.

As P increases the Amdahl limit for parallel speed-up becomes increasingly apparent. To achieve profitable speed-up with large P it is therefore important to ensure best possible load balance for all parallel loops. With inter-processor communication performance continuing to improve it is anticipated that secondary partition load balancing will continue to be a profitable exchange for data communication. Optimisation of partition realignment remains a difficult problem, especially as the misalignment of partitions becomes significant. A current focus is apply JOSTLE to the solution of this optimisation problem and to incorporate partition alignment rules into the dynamic load balancing strategies being developed for PHYSICA.

Acknowledgements
The authors wish to thank HLRS, the High Performance Computer Centre at the University of Stuttgart for access to the Cray T3E.

REFERENCES

1. M. Cross. Computational issues in the modelling of materials based maufacturing processes. *J. Computer Aided Materials Design*, 3:100–116, 1996.
2. M. Cross, C. Bailey, K. A. Pericleous, K. McManus, S. Bounds, G. Moran, G. Taylor, and D. Wheeler. Multi-physics modelling – a vital component of virtual manufacturing. Agard-r-821, NATO Advisory Group for Aerospace Research & Development, May 1998.
3. M. Cross, P. Chow, C. Bailey, N. Croft, J. Ewer, P. Leggett, K. McManus, and K. A. Pericleous. PHYSICA - a software environment for the modelling of multi-physics phenomena. In *Proc ICIAM 1995*, 1996.
4. K. McManus, C. Walshaw, M. Cross, P. Leggett, and S. Johnson. Evaluation of the JOSTLE mesh partitioning code for practical multiphysics applications. In A. Ecer, J. Periaux, N. Satofuka, and S. Taylor, editors, *Parallel Computational Fluid Dynamics, implementations and results using parallel computers*, pages 673–680, 1996. Proc Parallel CFD 1995.
5. D. Vanderstraeten, R. Keunings, and C. Farhat. Beyond Conventional Mesh Partitioning Algorithms and the Minimum Edge Cut Criterion: Impact on Realistic Applications. In D. Bailey *et al*, editor, *Parallel Processing for Scientific Computing*, pages 611–614. SIAM, 1995.
6. C. Walshaw, M. Cross, R. Diekmann, and F. Schlimbach. Multilevel Mesh Partitioning for Optimising Domain Shape. Tech. Rep. 98/IM/38, Univ. Greenwich, London SE18 6PF, UK, July 1998.
7. C. Walshaw, M. Cross, and M. Everett. Mesh partitioning and load-balancing for distributed memory parallel systems. In B. Topping, editor, *Proc. Parallel & Distributed Computing for Computational Mechanics, Lochinver, Scotland, 1997*, 1997.