Proceedings of the IASTED International Conference
Parallel and Distributed Computing and Systems
November 3-6, 1999 in Cambridge Massachusetts, USA

Characterizing the Performance of Dynamic Distribution and Load-Balancing Techniques for Adaptive Grid Hierarchies

Mausumi Shee, Samip Bhavsar, and Manish Parashar
Department of Electrical and Computer Engineering, Rutgers University, 94 Brett Road, Piscataway, NJ 08854 Tel: (732) 445-5388; Fax: (732) 445-0593; Email: {mshee,samip,parashar}@caip.rutgers.edu

Abstract

Dynamically adaptive techniques for the solution of partial differential equations that employ locally optimal approximations can yield highly advantageous ratios for cost/accuracy. Distributed implementations of these methods offer the potential for accurate solutions of physically realistic models of important physical systems. These implementations however, lead to interesting challenges in dynamic data-distribution, re-distribution and load balancing. This paper presents an application-centric performance characterization and evaluation of dynamic partitioning and load-balancing techniques for distributed adaptive grid hierarchies that underlie adaptive mesh-refinement algorithms (AMR). The overall goal of this characterization is to enable the selection of the most appropriate mechanism based on application and system parameters.

Keywords: Dynamic load balancing; Performance characterization; Adaptive mesh refinement.

1. Introduction

This paper presents an application-centric performance characterization of dynamic partitioning and load-balancing techniques for distributed adaptive grid hierarchies that underlie parallel adaptive mesh-refinement (AMR) techniques for the solution of partial-differential equations. The goal of this characterization is to enable the selection of the most appropriate mechanism based on application and system parameters.

Dynamically adaptive methods for the solution of partial differential equations that employ locally optimal approximations can yield highly advantageous ratios for cost/accuracy when compared to methods based upon static uniform approximations. These techniques seek to improve the accuracy of the solution by dynamically refining the computational grid in regions of high local solution error. Distributed implementations of these methods offer the potential for accurate solution of physically realistic models of important physical systems. We believe that the next generation simulations of complex physical phenomenon will be built using such dynamically adaptive techniques executing on distributed heterogeneous computational grids, and will provide dramatic insights into complex systems such as interacting black holes and neutron stars, formations of galaxies, oil reservoirs and aquifers, and seismic models of the whole earth.

Distributed implementations of adaptive applications lead to interesting challenges in dynamic resource allocation, data-distribution and load balancing, communications and coordination, and resource management. The overall efficiency of the adaptive algorithms is limited by the ability to partition the underlying data-structures at run-time to expose all inherent parallelism, minimize communication and synchronization overheads, and balance load. A critical requirement while partitioning adaptive grid hierarchies that underlie these algorithms is the maintenance of logical locality, both across different levels of the hierarchy under expansion and contraction of the adaptive grid structure, and within partitions of grids at all levels when they are decomposed and mapped across processors. The former enables efficient computational access to the grids while the latter minimizes the total communication and synchronization overheads. Furthermore application adaptivity results in application grids being created, moved and deleted on the fly, making it necessary to efficiently re-partition the hierarchy on the fly so that it continues to meet these goals.

This paper first presents metrics for an application-centric performance characterization of distribution mechanism for AMR grid hierarchies. It then uses them to characterize the performance of a suite of partitioning and load-balancing mechanisms used by distributed AMR infrastructures. The Buckley-Leverette applications kernel from oil reservoir simulation applications is used to evaluation the different schemes. The evaluation is performed in an architecture independent manner. This enables selection of the most appropriate partitioning/load-balancing scheme for a specific architecture by matching the metrics against the system parameters. The rest of this paper is organized as follows: Section 2 describes the family of adaptive algorithms targeted in this paper. Parallel and distributed implementations of these are discussed. Section 3 presents an application-centric performance characterization of distribution and load-balancing techniques for adaptive applications. Section 4 outlines a suite of six distribution and load-balancing schemes used by current infrastructures supporting adaptive applications. Section 5 uses a real-world AMR application to characterize the distribution and load-balancing schemes. Section 7 presents our conclusions and outlines future work.

2. Problem Description

Dynamically adaptive numerical techniques for solving differential equations provide a means for concentrating computational effort to appropriate regions in the computational domain. In the case of hierarchical adaptive mesh refinement (AMR) methods, this is achieved by tracking regions in the domain that require additional resolution and dynamically overlaying finer grids over these regions. AMR-based techniques start with a base coarse grid with minimum acceptable resolution that covers the entire computational domain. As the solution progresses, regions in the domain with high solution error and requiring additional resolution are tagged and finer grids are overlaid on the tagged regions of the coarse grid. Refinement proceeds recursively so that regions on the finer grid requiring more resolution are similarly tagged and even finer grids are overlaid on these regions. The resulting grid structure is a dynamic adaptive grid hierarchy. Figure 1 shows the adaptive grid hierarchy for the Berger/Oliger AMR formulation.

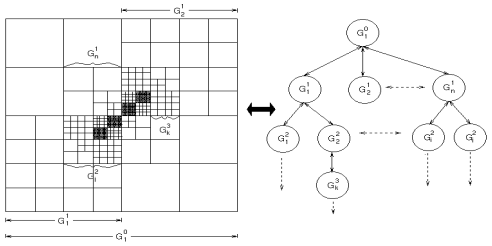


Figure 1 – 2D Adaptive Grid Hierarchy

Distribution of adaptive methods based on hierarchical AMR consists of appropriately partitioning the adaptive grid hierarchy across available computing nodes, and operating on the local portions of this domain in parallel. Parallel AMR implementations primarily require two types of communication:

1. Inter-grid Communications: Inter-grid Communications are defined between component grids at different levels of the grid-hierarchy and consist of *prolongations* (coarse to fine transfers) and *restrictions* (fine to coarse transfers). These communications typically require a gather/scatter type operation based on an interpolation or averaging stencil. Inter-grid communications can lead to serialization bottlenecks for naïve decompositions of the grid hierarchy.
2. Intra-grid Communications: Intra-grid Communications are required to update the grid-elements along the boundaries of local portions of a distributed grid. These communications consist of near-neighbor exchanges using the stencil defined by the finite-difference operator. Intra-grid communications are regular and can be scheduled so as to overlap with computations on the interior region of the local portions of a distributed grid.

Note that on the same processor, these communications translate to memory copies. The key requirements for a decomposition scheme for partitioning an adaptive grid hierarchy can be summarized as: (1) expose available data-parallelism (2) minimize communication overhead (3) balance overall load distribution and (4) enable dynamic load redistribution with minimum overheads. A balanced load distribution is particularly critical for parallel AMR based applications as different levels of the hierarchy have different computational loads. Furthermore, for time dependent simulations, the grid hierarchy changes with grids being created, moved and destroyed at different time steps, making efficient load-redistribution critical**.**

3. A Characterization of AMR Distribution Mechanisms

We use four sets of metrics to characterize distribution and load-balancing mechanisms for AMR adaptive grid hierarchies, viz. load balance, distribution quality, grid interaction overheads (inter-processor communication and memory copy), and data-movement overheads. These metrics are described below.

3.1. Load Balance

The load balance metric measures the distribution of load across the processors and the time taken to achieve the distribution. Balanced load distribution is particularly critical for parallel AMR applications as different levels of the hierarchy have different computational loads. Furthermore, AMR applications require re-distribution and load balancing at regular intervals; consequently the time spent in this effort is critical. The goal of this metric is to define operational points that represent the best balance between the effort spent in balancing the load and the balance achieved.

3.2. Distribution Quality

Distribution quality is quantified by the number of grid components created on each processor and the quality (size, aspect ratio) of these components. The former captures the overheads due to the allocation, management and de-allocation of grid components. Large number of small grids also increases the number of memory copies required for inter-level and intra-level communications. The size and shape of the grids also affects the communication/memory copy behavior. Bad aspect ratios result in larger interfaces between sibling grids and increased intra-level communications. Finally grid size also affects the overall cache behavior. Our goal is to use this metric to determine an acceptable range for the shape and size of grid components for different architectures, and use this to drive the distribution. In this paper we evaluate the number of boxes created for each partitioning/load-balancing scheme studied.

3.3. Grid Interaction Overheads

The grid interaction overhead metric aims at characterizing the ability of the distribution scheme to capture and maintain application locality. Here we measure the overheads of four kinds of communications: inter-grid communications between grids at different levels, intra-grid communication along ghost boundaries,

and inter- and intra grid memory copies for co-located grid components. Maintaining locality to minimize these overheads can lead to conflicting optimizations. The objective of this metric is to identify a balance between the two overheads based on system memory architecture and communication characteristics that can achieve best overall performance.

3.4. Data Movement

Every refinement step in the AMR algorithms typically causes the adaptive grid hierarchy to change requiring load balancing and data-redistribution. Redistribution should be incremental so as to minimize the data that has to be relocated. The objective of the data movement metric is to characterize the ability of the distribution scheme to minimize redistribution costs by reassigning grids to their original location. Optimizing this metric can lead to conflicts with requirements for optimizing load balancing and interaction overheads.

4. Run-Time Partitioning Dynamic AMR Grid Hierarchies

This section describes the six dynamic partitioning and load balancing schemes that we have implemented and evaluated in this paper. These schemes underlie five existing AMR infrastructures: BATSRUS[1], PARAMESH[6], SCOREC PMDB[8], SAMRAI[7] and DAGH[2].

4.1. Space-Filling Curves

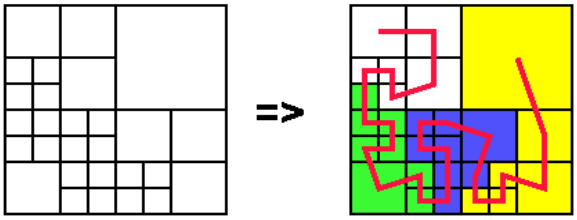


Figure 2 - Space-Filling Curve Representation of an Adaptive Grid Hierarchy

Space-filling curves (SFC) [4] are a class of locality preserving mappings from d-dimensional space to 1-dimensional space i.e., $N^d \rightarrow N^1$, such that each point in N^d is mapped to a unique point or index in N^1. The self-similar or recursive nature of these mappings can be exploited to represent a hierarchical structure and to maintain locality across different levels of hierarchy. The SFC representation of the adaptive grid hierarchy is a 1-D ordered list of composite grid blocks where each composite block represents a block of the entire grid hierarchy and may contain more than one grid level; i.e. inter-level locality is maintained within each composite block. Figure 2 illustrates the composite representation for a two-dimensional grid hierarchy. Using the space-filling curve representation, the adaptive grid hierarchy can be simply partitioned by partitioning the composite list to balance the total work assigned to each processor. This decomposition using the Peano-Hilbert space-filling ordering for a 1-D grid hierarchy is shown in Figure 3. As inter-level locality is inherently maintained by the composite representation, the decomposition generated by partitioning this representation eliminates expensive gather/scatter communication and allows prolongation and restriction operations to be performed locally at each processor.

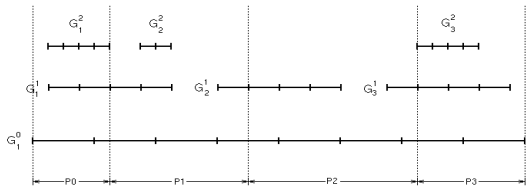


Figure 3 - Space-Filling (Composite) Distribution

4.2. Independent Grid Distribution

The independent grid distribution (IGD) scheme, shown in Figure 4, distributes the grids independently across the processors. This distribution leads to balanced loads and no redistribution is required when grids are created or deleted. However the decomposition scheme can be very inefficient with regard to inter-grid communication. In the adaptive grid hierarchy, a fine grid typically corresponds to a small region of the underlying coarse grid. If both, the fine and coarse grid are distributed over the entire set of processors, all the processors will communicate with the small set of processors corresponding to the associated coarse grid region, causing a serialization bottleneck.

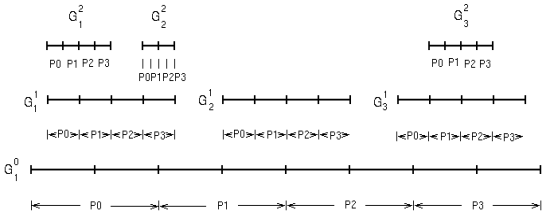


Figure 4 - Independent Grid Distribution

4.3. Combined Grid Distribution

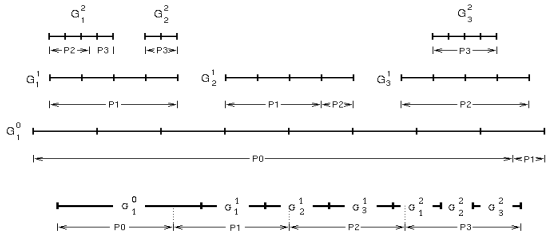


Figure 5 - Combined Grid Distribution

The combined grid distribution (CGD), shown in Figure 5, distributes the total work load in the grid hierarchy by first forming a simple linear structure by abutting grids at a level and then decomposing this structure into partitions of equal load. The combined decomposition scheme also suffers from the serialization bottleneck described above but to a lesser extent. For example, in Figure 5, G21 and G22 update G11 requiring

P2 and P3 to communicate with P1 for every restriction. Regriding operations involving the creation or deletion of a grid are extremely expensive in this case, as they require an almost complete redistribution of the grid hierarchy. The combined grid decomposition does not exploit the parallelism available within a level of the hierarchy. For example, when G01 is being updated, processors P2 and P3 are idle and P1 has only a small amount of work. Similarly when updating grids at level 1 (G11, G12 and G13) processors P0 and P3 are idle, and when updating grids at level 2 (G21, G22 and G23) processors P0 and P1 are idle.

4.4. Independent Level Distribution

In the independent level distribution (ILD) scheme (see Figure 6), each level of the grid hierarchy is distributed by partitioning the combined load of all component grids at the level among the processors. This scheme overcomes some of the drawbacks of the independent grid distribution. Parallelism within a level of the hierarchy is exploited. Although the inter-grid communication bottleneck is reduced in this case, the required gather/scatter communications can be expensive. Creation or deletion of component grids at any level requires a redistribution of the entire level.

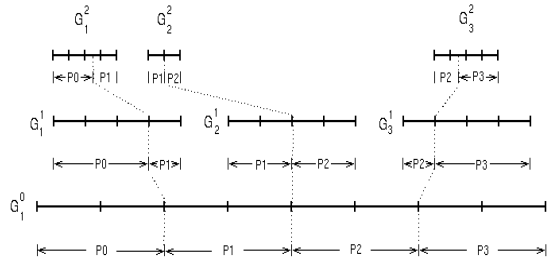


Figure 6 - Independent Level Distribution

4.5. Iterative Tree balancing

The iterative tree balancing (ITB) scheme treats the dynamic partitioning and load-balancing problem as a graph-partitioning problem. A table is created from the grids at each time step, which keeps pointers to neighboring and parent grids. A breadth first search is made on this graph i.e. for every grid, immediate neighbors and children are also considered along with load distribution. Thus load balancing, inter level communication and intra level communication are addressed together. This scheme is used for distributing fine-element meshes and is promising as it deals with all the constraints to some extent.

4.6. Weighted Distribution

The weighted distribution scheme is a heuristic based hybrid scheme that attempts to combine the features of the other schemes described in this section. As previously observed, there are three primary parameters that need to be controlled to minimize the overheads of an adaptive grid hierarchy distribution, viz. intra-level communication, inter-level communication and data movement at each regrid. In the weighted distribution scheme we first assign a weight to each of these overheads. This weight defines the significance and contribution of the overhead to the overall application performance and depends on the system architecture and dynamic application behavior. The next step uses these weights to compute the affinity of each component grid to the different processors. Initially, grids have no affinity for any processor. For each grid, the affinity of the processor(s) housing its parents is now increased by the inter-level communication weight. Similarly the affinities of processors housing the neighbors of the grid are increased by the intra-level communication weight, and affinity of the original location of the grid is increased by the data-movement overhead weight. The grid is now assigned to the available processors (i.e. total assigned load is below threshold for load balancing) to which the grid has maximum affinity. If the grid has equal affinity to more than one processor, the grid is either split among the processor (if its size is greater than the size threshold) or assigned to the processor with least load. For example if the application has many component grids and uses a large stencil, then the dominating weight is associated with intra-level communication. Similarly if the application is very dynamic and needs to regrid very often, the data-movement weight dominates. Weights assigned to the different parameters can change dynamically depending on the current application and system states.

5. Experimental Evaluation

To evaluate the six distribution/load-balancing schemes outlined in Section 5 we use a grid adaptation trace from the 3D Buckley-Leverette equation kernel (BL) used in oil reservoir simulations. This application simulates the flow of the oil pressure front across the oil field between wells. Grid refinements in the application track the front moving diagonally across the grid. The adaptation trace was generated from a single processor run with 5 levels of factor 2 refinement. The BL trace consisted of 66 regrid steps with regridding every fourth time step. The base (coarsest) grid was 33^3. The trace was then fed to a partitioning module that partitioned the boxes across the required number of processor using each of the six schemes. An AMR simulator then evaluated various costs for the partition generated. The evaluation was performed using architecture independent measurements. These measurements for the different metrics are summarized in the table below. In case of communication and data-movement overheads, this corresponds to the amount of information communicated. In case of distribution quality metrics this corresponds to the number of grids per processor and the percentage load imbalance (a perfect load balance corresponds to 0%). The motivation for keeping the measurement architecture independent was to enable them to be used to select appropriate schemes for a wide range of architecture using a very simple architectural description.

Metric	Measurement

Load Balance	Percentage load imbalance
Intra-level/Inter-level communication overhead	Megabytes communicated
Intra-level/Inter-level memory copy	Megabytes copied
Data movement	Megabytes moved
Distribution time	Seconds
Distribution quality	Number of Boxes (x 1000)

The plots presented in this section represent cumulative costs for each scheme. The vertical axis in each of these plots is the relevant measurement for each metric, while the horizontal axis is the number of processors. In this experimentation we used three configurations of 16, 32 and 64 processors. The plots in Figure 9 show the intra and inter level communication overheads, while Figure 10 show the memory copies in each case. In both the sets of plots, the IGD distribution scheme tends to be particularly bad. The other schemes tend to be comparable with the SFC and CGD being slightly better than the rest. The plots in Figures 11 and 12 show the overheads due to dynamic distribution and load balancing. Once again IGD results in the maximum data-movement while CGD has the least data movement. The ILD scheme requires the maximum time for dynamic distribution and load balancing. Finally the plots in Figure 13 show the quality of distribution in each case. Once again IGD results in the maximum number of boxes, however, it results in a near perfect load balance. ITB produces the overall best quality distribution with SFC and CGD schemes comparable. ILD produces the most load imbalance.

6. Conclusions and Future Work

In this paper we presented a performance characterization of six dynamic partitioning and load-balancing techniques for distributed adaptive grid hierarchies that underlie parallel adaptive mesh-refinement (AMR) techniques for the solution of partial-differential equations. This work is part of an ongoing project for developing policy driven "smart" tools for automated distribution/load balancing of adaptive application in heterogeneous distributed environments. The characterization presented consists of 3 metrics: Interaction overheads (inter- and intra- level communications and copies), Distribution Quality (load-balance, number of grids) and Distribution Overheads (data movement and distribution/load balancing time). The Buckley–Leverette AMR kernel from oil reservoir simulation applications is used to experimentally evaluate the distribution/load-balancing schemes. The presented results show that the iterative tree-balancing scheme provides best overall performance for this application with the space-filling curve and weighted distribution providing comparable distributions. The reason is that these techniques use application information to determine the partitioning rather than pure heuristic. We are currently expanding this characterization to other applications and encoding the results into a policy rule base that can drive an automated partitioning and load-balancing tool.

7. References

[1] BATSRUS: hpcc.engin.umich.edu/HPCC/codes\\/2/BATSRUSv2.html.

[2] Distributed Adaptive Grid Hierarchies, www.caip.rutgers.edu/~parashar/DAGH/.

[3] H. Sagan, Space-Filling Curves. Springer-Verlag, 1994.

[4] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley Company, 1989.

[5] M. Parashar and J.C. Browne, "On Partitioning Dynamic Adaptive Grid Hierarchies", Proceedings of the 29th Annual Hawaii International Conference on System Sciences, Jan. 1996.

[6] PARAMESH: sdcd.gsfc.nasa.gov/ESS/eazydir/\\inhouse/macneice/paramesh/paramesh.html.

[7] SAMRAI: Structured Adaptive Mesh Refinement Applications Infrastructure, www.llnl.gov/CASC/SAMRAI/.

[8] SCOREC Parallel Scientific Computation: www.scorec.rpi.edu/programs/parallel/ParallelScientific.html

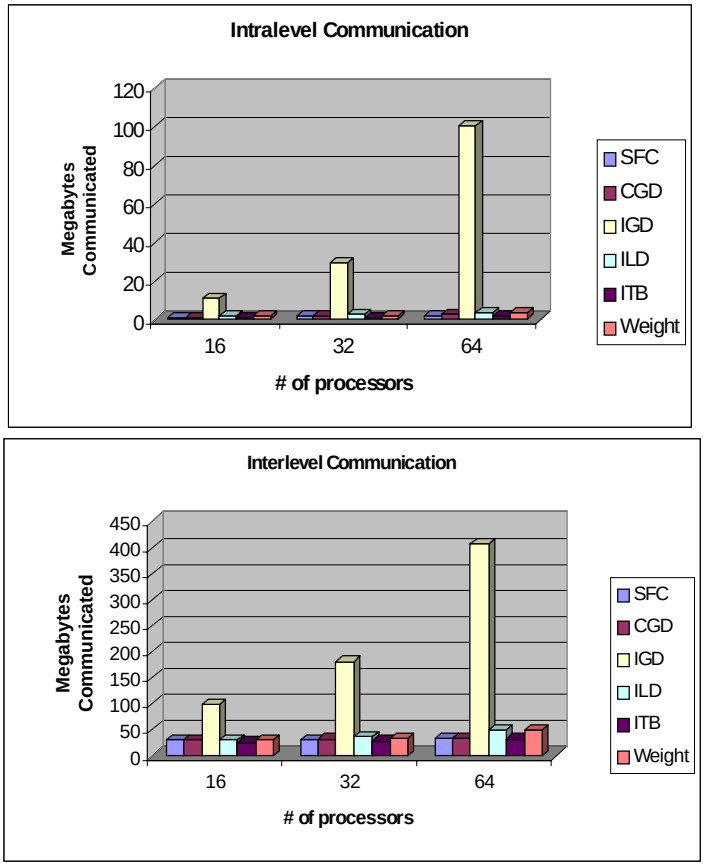
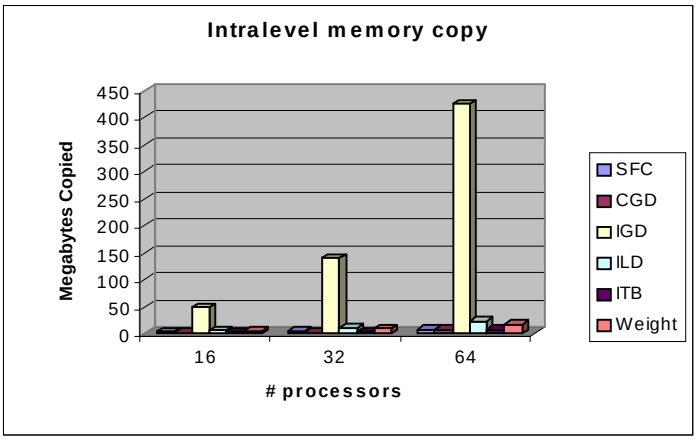


Figure 9 – BL: Communication Overheads



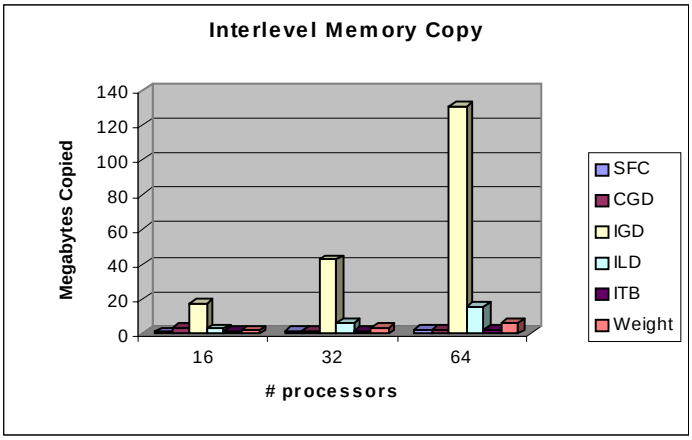


Figure 10 – BL: Memory Copy Overheads

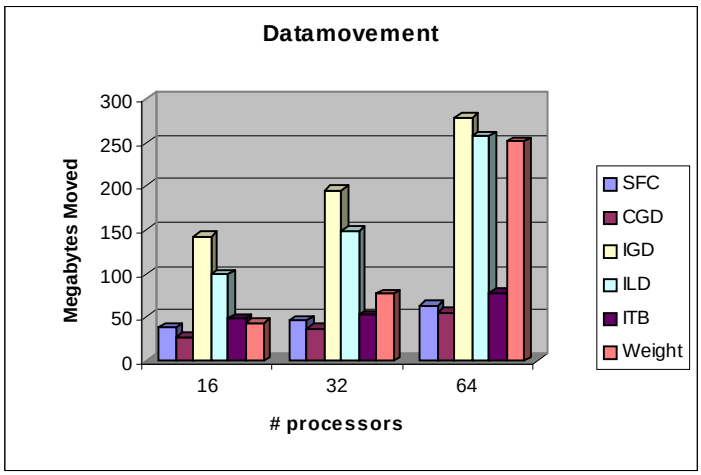


Figure 11 – BL: Data Movement Overhead

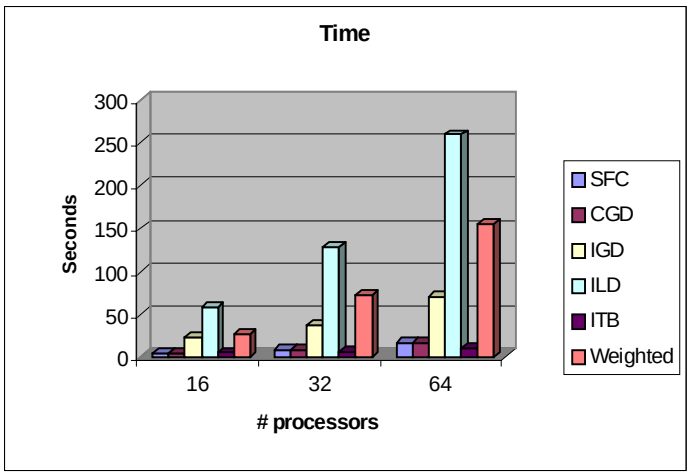


Figure 12 – BL: Distribution/Load-Balancing Time

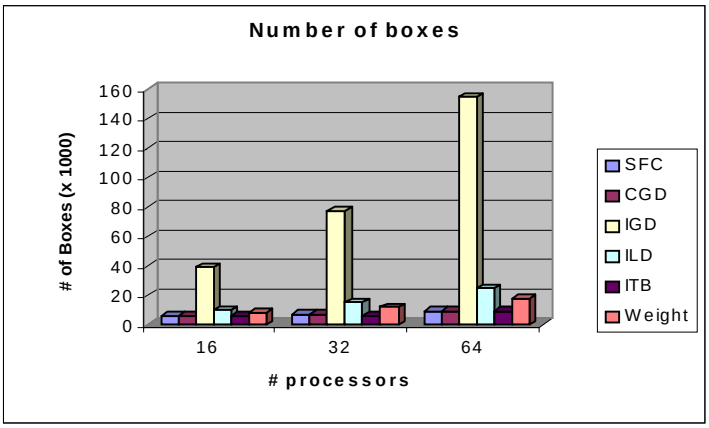


Figure 13 – BL: Distribution Quality

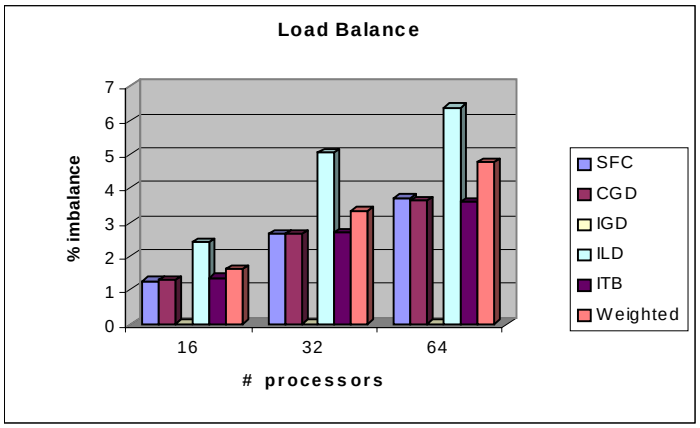


Figure 14 – BL: Distribution Quality