6. Parallel Volume Rendering Algorithms

This chapter introduces a taxonomy of parallel volume rendering algorithms. In the thesis statement we claim that parallel algorithms may be described by "... *how the tasks and data are partitioned over the architecture*." The taxonomy of algorithms is based on how the image and object lattices are distributed. We assume a "generic" MIMD parallel system based on compute nodes with local memory and a communications network. Communication costs for the algorithms are based on the quantity of data that moves over the network every frame.

The rendering method is part of the task branch of the taxonomy. Many parallel algorithms permit the use of either image or object order rendering methods. With such algorithms, the choice of rendering method becomes important when considering the communications model of the actual system. Chapter seven deals with this choice and other issues of practical implementation.

The notation used in the remaining chapters is an extension of that used in chapter five. The terminology will be explained as it is introduced.

n	number of nodes in the system
d	number of points in the object lattice
p	number of screen pixels
m_{redist}	quantity of data communicated in redistribution
t_{redist}	time cost of redistribution
ϕ	rotation angle(s) of image lattice axes from object lattice axes
b_{bis}	bisection bandwidth
b_{link}	communication-channel bandwidth

6.1. Taxonomy

The full design-space of algorithms is illustrated in figure 6.1. The primary distinction is between *image partitions* and *object partitions*. A partition is named for the lattice onto which computing tasks are mapped. In an image partition, a node's task is to compute a subset of image-lattice points while object-lattice points are communicated as required by the view. In an object partition, each node renders an image of its local object-lattice subset, and communicates the resulting image as necessary for compositing.

For either partition, the image and object lattices must be distributed among the nodes. In chapter one, we defined three subsets of a volume: *slabs, shafts,* and *blocks*. The image and object lattices are distributed as one of these. If lattice subsets are more numerous than the number of nodes, some nodes have more than one subset of a lattice. Multiple lattice subsets at a node may be spatially adjacent and together form a larger, *contiguous*

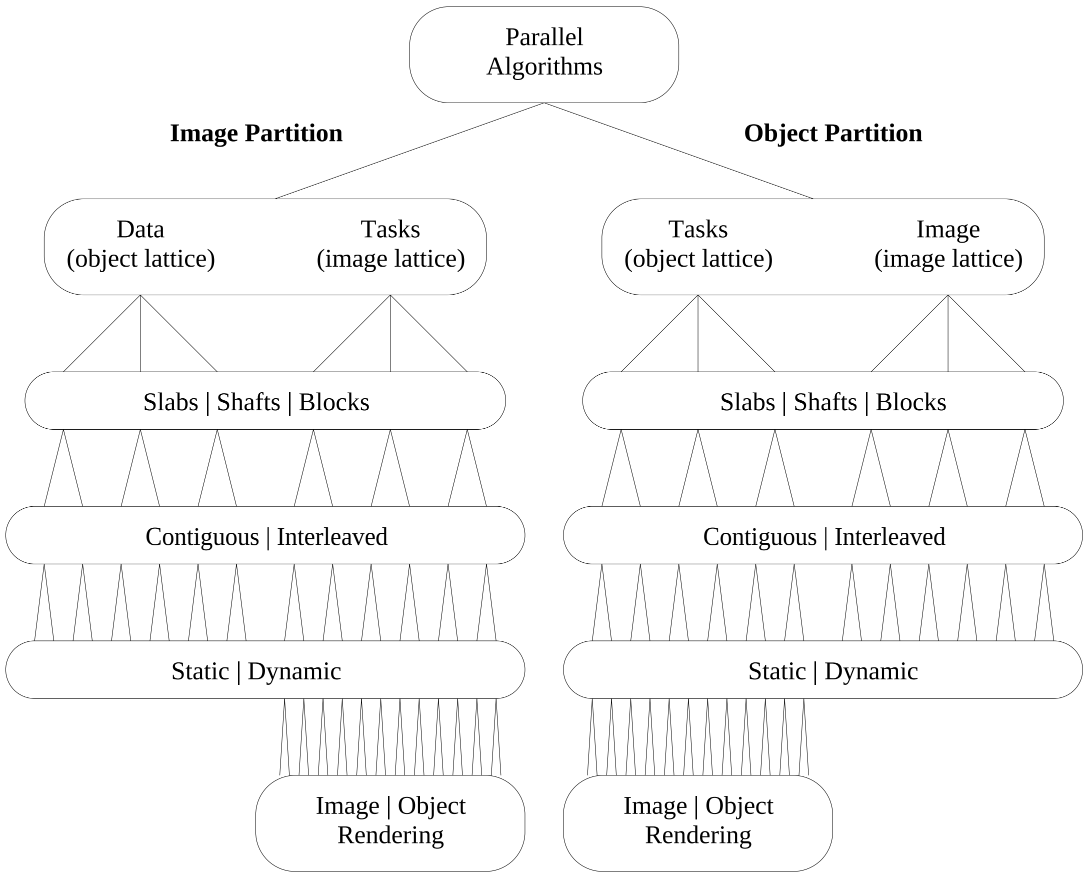


Fig. 6.1 - Full taxonomy of parallel volume-rendering algorithms

subset of the lattice, or they may be spatially-separate subsets which we will refer to as *interleaved*. If the lattice subsets at a node remain the same over time, the lattice distribution is *static*. If the subsets vary from frame to frame, or within a frame, the distribution is *dynamic*.

Optimizing the distribution of the lattices among the memory and computing resources in the system is important. The distribution should make efficient use of scarce or expensive resources. In parallel systems today, the communications network is the most expensive resource to use if we define "expensive" in terms of time. Current networks are many times slower than paths to main memory. An efficient algorithm, therefore, is one which minimizes the communication requirement imposed on the network. The communication of lattice subsets is referred to as *redistribution* to distinguish it from other communication such as control messages. Trivial distributions like fully replicating the data set at each node are deemed too expensive as a general solution, although some replication of lattice points is often necessary or desirable. Some lattice distributions in the taxonomy are inherently impractical or inefficient. These will be culled in the remaining sections, leaving the remaining algorithms for consideration in chapter seven. Communication requirements for these algorithms will be derived and used in chapter seven as a basis for gauging their relative efficiencies.

6.2. Image Partitions

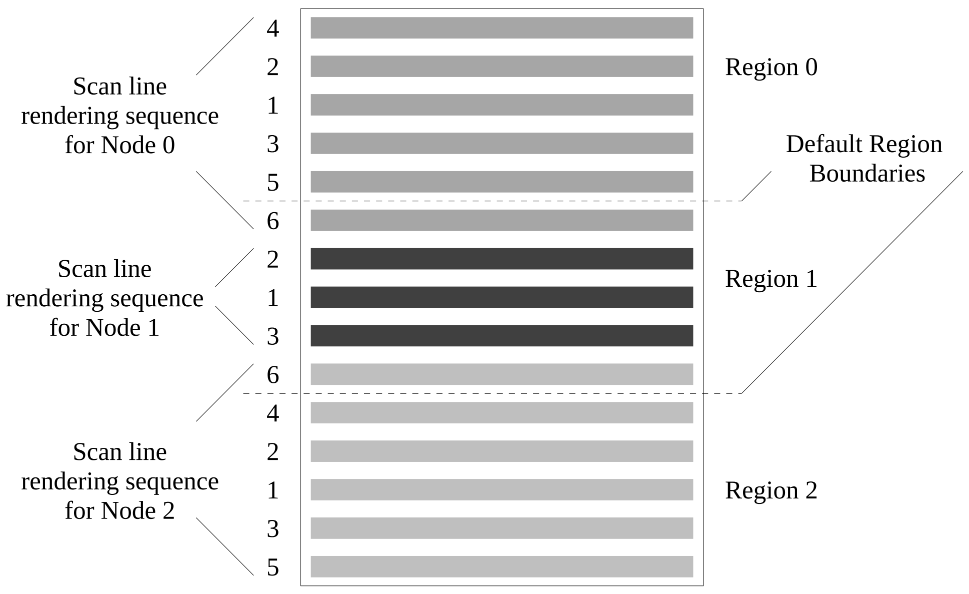
Image-partition algorithms start each frame with the redistribution phase. Rendering tasks are distributed among the nodes by assigning them subsets of the image lattice. For example, a node assigned an image-lattice slab would render a horizontal or vertical screen-stripe; a node assigned an image-lattice shaft would render a rectangular screen area. For convenience, any subset (rectangular or irregular) of screen pixels will be refered to as a *region*. The resampling process must have access to all the object-lattice data points that fall into a node's assigned image-lattice subset under the view transformation. Object-lattice data subsets are also distributed among nodes. The view transformation determines where a node's object-lattice subsets are needed. Redistribution for image partitions is a screen-space sort of the transformed data points. The amount of data to redistribute is $m_{\text{redist}} = f(d, n, \phi)$, a function of data size, the number of nodes, and the view point. The redistribution size grows as the data size, even with the assumption that some fraction of the data set will be zeros and therefore not redistributed.

6.2.1. Image Lattice Distribution

In an image partition, the rendering task is associated with the image-lattice distribution. Distributing the image lattice as slabs or shafts makes each node responsible for rendering one or more rectangular regions of the final image. If the image lattice is distributed as blocks, a final composite must be done, adding additional communication beyond the redistribution cost. Because of this additional compositing cost, block distributions will not be considered in any greater detail here. (In chapter seven, block distributions are shown appropriate for 3D network topologies.) A contiguous distribution allows each node to render a single screen region. An interleaved distribution causes several separate regions to be rendered at each node; for example, scan lines (slabs) could be assigned to nodes in round-robin fashion. A distribution is static if it never changes. A dynamic distribution changes within or between frames.

6.2.1.1. Load Balancing

The load at each node is proportional to the number of image-lattice points to be computed. To balance the load, the image-lattice distribution must either be varied or be statistically equivalent at each node. It is not possible to balance the loads among processing nodes by using a contiguous static distribution - there is no way to make any adjustments. Load balancing may be performed with a contiguous dynamic distribution by varying the screen region boundaries. This approach is simplest with slab distributions. Figure 6.2 illustrates the advantage of rendering out from the center of the default region to allow growth or shrinkage of the slabs in both directions. Shaft distributions are more difficult to balance since the regions are difficult to fit together if balancing is done in two dimensions. Figure 6.3 illustrates the work-load for a frame



Scan line rendering sequence for Node 0
Scan line rendering sequence for Node 1
Scan line rendering sequence for Node 2
4
2
1
3
5
6
2
1
3
6
4
2
1
3
5
Region 0
Default Region Boundaries
Region 1
Region 2

Fig. 6.2 - Load balancing with dynamic contiguous slabs

balanced this way. Each node starts in the center of its default region and alternately steps out left and right until it hits both neighbor pixels, or the screen edge. This approach attempts to render whole scan lines in-step across shafts. Consecutive scan lines are rendered as in the slab distribution approach - out from the center. Image-order rendering is necessary with contiguous dynamic distributions since the image-region boundaries change adaptively. No implementation of a contiguous dynamic distribution has been reported.

Static or dynamic interleaved-distributions also support load balancing. Static interleaved distributions attempt to achieve load balance by statistically equalizing the loads. The image lattice is distributed as many small regions in round-robin fashion. The benefit of this approach is its simplicity and lack of additional computation cost. The

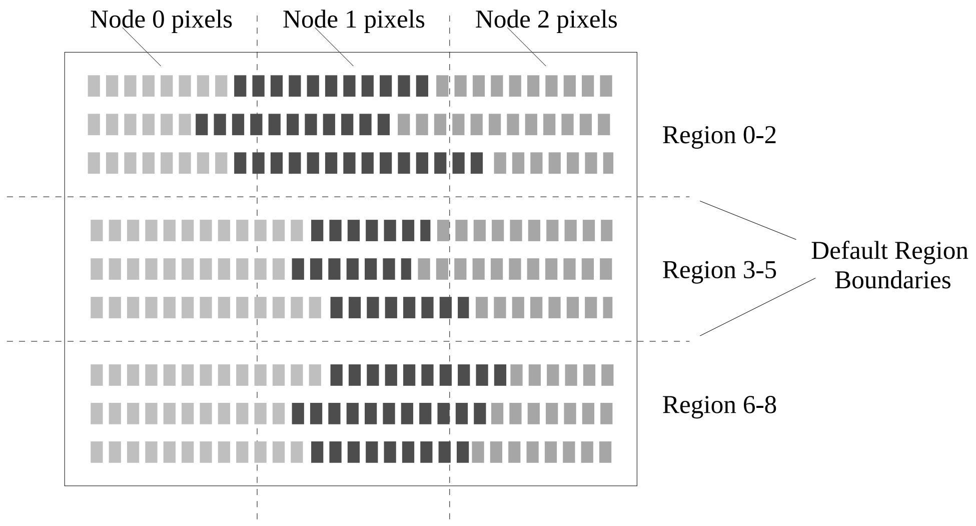


Fig. 6.3 - Dynamic contiguous shaft load balancing on scan lines

drawback is that to achieve good balance the region size must be small enough to assure that each node gets many regions. As the regions get smaller the redistribution size goes up. Many object-lattice points contribute to each image-lattice point due to the overlapping extents of the reconstruction filters. Object-lattice points that transform near region boundaries will be needed in multiple regions, causing many object lattice points to be communicated to multiple nodes. Static interleaved distributions have been implemented [Montani+92] [Westover91] using both image and object-order rendering methods.

Dynamic interleaved distributions may use coarse, even variable-size image lattice regions. This approach trades a lower redistribution penalty for an adaptive balancing method with some computing cost and control-message traffic. Dynamic interleaved distributions have been implemented [Challenger91] [Nieh+92] [Corrie+92] and shown to be effective at load balancing. Dynamic region-assignment requires image-order rendering since region size is adjusted within a frame.

Figure 6.4 illustrates the image-partition task-distribution options with the impractical or inefficient options diminished.

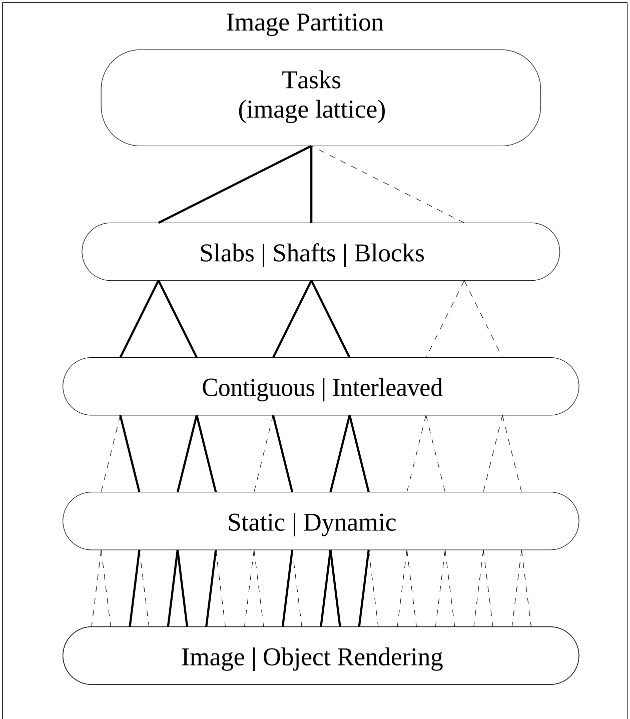


Fig. 6.4 - Image-partition task distributions

6.2.2. Object Lattice Distribution

In an image partition, the data distribution is the primary determinant of the redistribution cost. Object-lattice data moves over the network at the granularity of the distribution. Slabs and shafts have irregular aspect ratios and are too coarse to be efficient. Blocks are the only practical object-lattice subset. The size of the blocks is the smallest number of lattice points moved in a single, atomic communication *event*. An event is a single message in a message passing system or a single remote-access transfer in a shared memory system. Block size is adjustable in message passing systems but usually not in shared memory systems. The block granularity sets the trade-off between efficient network utilization and the number of communications events.   Fine granularity blocks cause a high number of communication events, but the number of excess lattice points moved to nodes that do not need them is low. With coarse blocks, there are fewer communications events, but a larger number of unnecessary points are moved.

6.2.2.1. Static Distribution

In a static data distribution, object-lattice blocks are permanently assigned to nodes. If there are multiple blocks per node they may be contiguous or interleaved. A contiguous data distribution coupled with a contiguous image-lattice distribution gives rise to a large variance in the redistribution occurring at each node. Under some view-transformations, a node's data will be required in its assigned image lattice, causing little or no data to be redistributed. Under other transformations, there will be no overlap of lattices, causing maximum redistribution. For any static contiguous data distribution, there are worst-case views for which all data is redistributed.

If either lattice is interleaved, the variance of the redistribution size is lowered. As noted in section 6.2.1.1, redistribution size increases when the image lattice is interleaved. However, interleaving the object lattice in an image partition does not increase redistribution size. Interleaving the data is also desirable since it randomizes the redistribution accesses. As the granularity of interleaved blocks becomes finer, redistribution becomes view independent. Jason Nieh's implementation [Nieh$^+$92] combines a dynamic interleaved image-lattice and a static interleaved data distribution.

A special case arises if the view point is limited to rotations about a subset of axes, and scaling and translation are restricted. A static-slab data and task distribution can avoid all redistribution costs by splitting the lattices in the plane perpendicular to the axis of rotation. Limited three-axis rotation may be achieved by replicating the data three times and storing slabs in all three axis-orientations at each node. Perspective is even possible if some overlap of data at the split planes is maintained. This approach severely constrains the possible view points but may be acceptable in some applications.

6.2.2.2. Dynamic Distribution

Data blocks in a dynamic distribution migrate among the nodes in response to view-transformation changes. There are three potential advantages to this approach over the static data distribution:

1. By limiting the change in view point from frame to frame, the redistribution size is bounded, potentially far below the worst case encountered with static distribution.
2. No redistribution is done when only shading or classifying parameters are changed.
3. It is possible to use only nearest-neighbor communication for redistribution in 2D and 3D mesh topologies.

The third advantage is only achieved if the image lattice is a dynamic contiguous distribution, or a static interleaved distribution. The neighbor relations of the nodes and their image-lattice subsets can not change as in a dynamic interleaved image-lattice distribution. Since data blocks migrate to image lattice subsets, an interleaved image-lattice distribution produces an interleaved data distribution while a contiguous image-lattice distribution produces a contiguous data distribution.

There is no reported implementation of an image partition using a dynamic data distribution. The low redistribution size of a dynamic distribution is approached by combining a static data distribution and a large data-cache [Corrie$^+$92]. As large caches and operating-system support for dynamically-migrating data appears in commercial multicomputers [Kendall], dynamic data distributions become easier to implement and are likely to become popular.

Figure 6.5 illustrates the generally useful image-partition data distributions.

6.2.3. Combined Image Partition Distributions

Figure 6.6 illustrates the useful image-partition options for combining the lattice distributions and rendering methods. There are seven reasonable possibilities - two using dynamic data distributions and five using static data distributions. In chapter seven, redistribution costs and implementation issues are examined for these approaches.

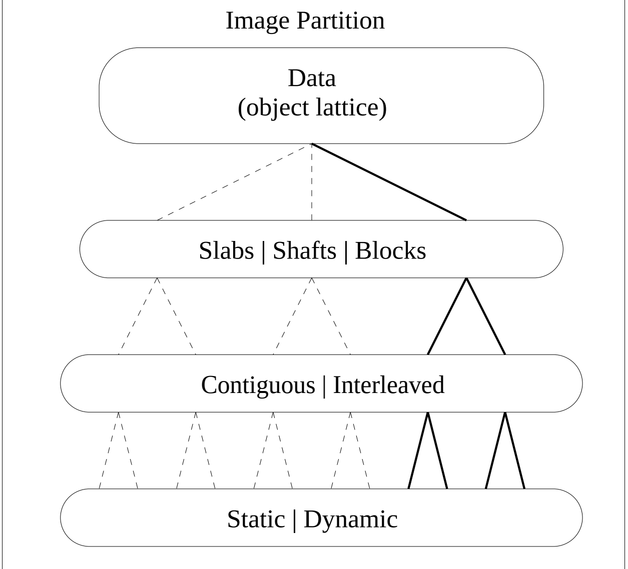


Fig. 6.5 - Image-partition data distributions

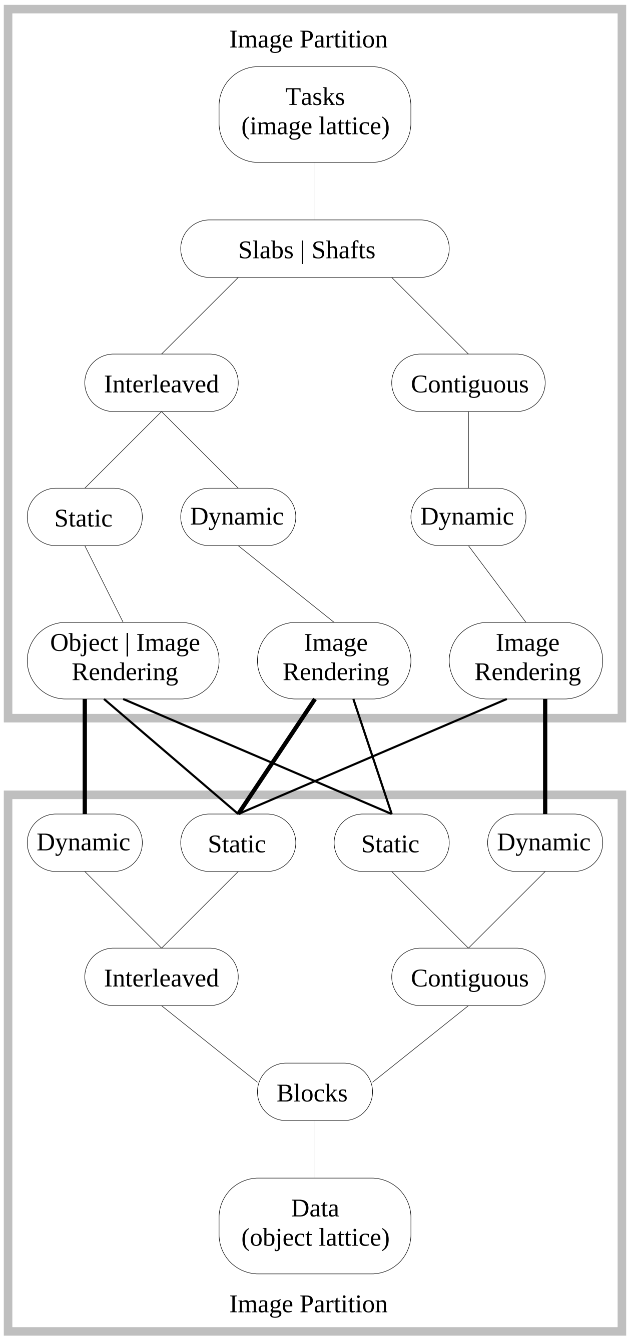


Fig. 6.6 - Image-partition options

6.3. Object Partitions

An object-partition algorithm redistributes image-lattice data. The reconstruction and resampling tasks are performed with only the local data at each node - object-lattice data is not communicated. Nodes compute the image-lattice points that fall in their local data-subsets under the view transformation. After resampling, image-lattice data is redistributed among the nodes to facilitate compositing. In an efficient object partition, locally computed image-lattice points are shaded, segmented, and composited before redistribution to minimize communication cost. Since rendering uses only local data, either object or image-order rendering methods may be used. Compositing is associative so it may be performed in two stages - a local pre-redistribution stage and a post-redistribution stage. The latter composites can only be performed after redistribution brings all the necessary data to a node. By using two-stage compositing, redistribution is done for 2D images rather than 3D volumes of data. Nodes render and redistribute both color and opacity images of their data subset(s) to facilitate the post-redistribution composite.

6.3.1. Object Lattice Distribution

In an object partition, the task distribution is determined by the object-lattice distribution among the nodes. Contiguous distributions are desirable since the number of object-lattice subsets is proportional to the redistribution size. Interleaved slabs, for example, would require multiple (possibly overlapping) images at each node to be redistributed; a single, contiguous slab produces only one image for redistribution.

Redistribution size for object partitions is $m_{\text{redist}} = f(n, p, \phi)$, a function of the number of nodes, the image size, and the view-point; the data size is not explicitly a parameter. The view-point affects the redistribution size as a function of the aspect ratio of the data distribution. Slabs, shafts, and blocks, in order, vary from very unbalanced aspect ratios to 1:1:1. Under rotation, these data subsets cover varying portions of the screen. In the worst case, slabs cover the complete screen, shafts run diagonally across the screen, and blocks are rotated 45 degrees to maximize their projected area. Of these, block distributions have the most-consistent redistribution size.

6.3.1.1. Load Balancing

Load balancing an object partition is accomplished by modifying the object-lattice subset size at each node, or making the subsets statistically equivalent. Static contiguous distributions do not allow load balancing since no adjustment mechanisms exist. Static interleaved distributions depend on a fine granularity of subsets to produce a statistically equivalent work load at each node. Recall that an interleaved distribution has a higher redistribution size due to the rendering of multiple images at each node. The only remaining efficient option is a dynamic contiguous distribution. Load balance is achieved by modifying the size of the object lattice at each node in response to a work estimate. The estimate may be based on a partial rendering of the current frame or the total load of the previous frame. The boundaries between adjacent object lattice subsets are adjusted to transfer work in the direction of lower average work load. This adjustment is done in one dimension at a time. For slab distributions, only one dimension may be adjusted, two adjustable dimensions exist for shafts, and three for blocks. This is a variant of the class of Orthogonal, Monotonic, and Surjective (OMS) grid-computation load-balancing methods described by Edoardo Biagioni [Biagioni91]. This method does not achieve optimal balance, but significant improvement is obtained. Chapter eight details the first reported implementation of an object-partition algorithm using a dynamic contiguous distribution with load balancing. Figure 6.7 illustrates the useful object-lattice distribution options.

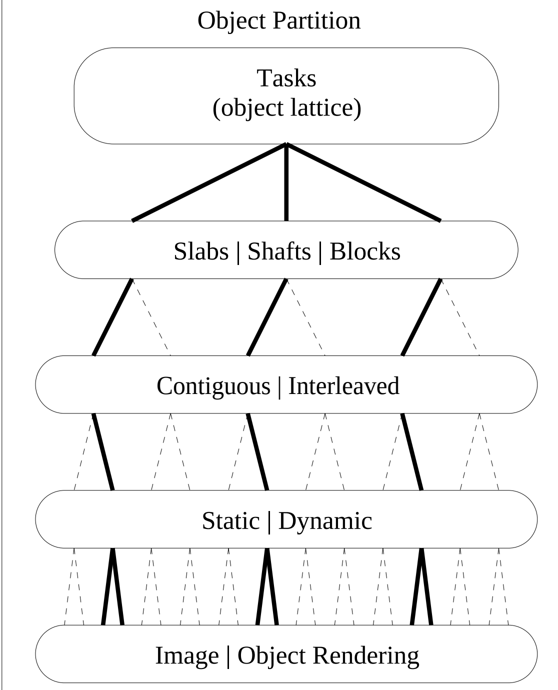


Fig. 6.7 - Object-partition task options

6.3.2. Image Lattice Distribution

The image of each node’s local data is redistributed to facilitate its compositing into the final image. Nodes are assigned portions of the screen for compositing. Since the image lattice is already reduced to two dimensional regions by local compositing, only slab and shaft distributions make sense to pursue. In a slab distribution, a node is responsible for compositing all local-image pixels on a set of scan lines. In a shaft distribution, a node composites all pixels for a rectangular region. Not all regions of the screen will have an equal number of local images to composite so there is some potential for load imbalance with a contiguous static distribution. Fortunately, the amount of work required to perform the post-redistribution compositing is usually small relative to the rendering work load, so the imbalance is not serious. A static interleaved distribution is a good choice for balancing the compositing load without introducing significant processing overhead. The overhead required to use a dynamic distribution may be too great to produce a net increase in performance. Figure 6.8 illustrates the useful image-lattice distribution options.

6.3.3. Combined Object Partition Distributions

Figure 6.9 illustrates that all the useful object-partition options for lattice distributions and rendering methods are compatible with each other. Chapter seven details the redistribution costs and implementation issues associated with these object partitions.

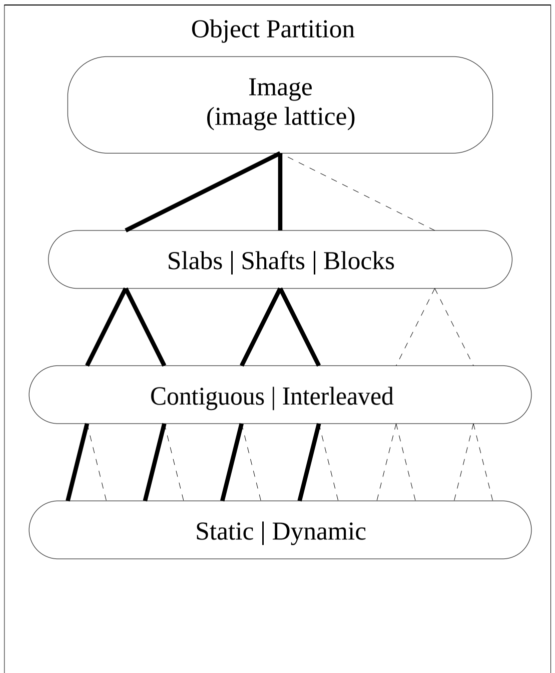


Fig. 6.8 - Object-partition image options

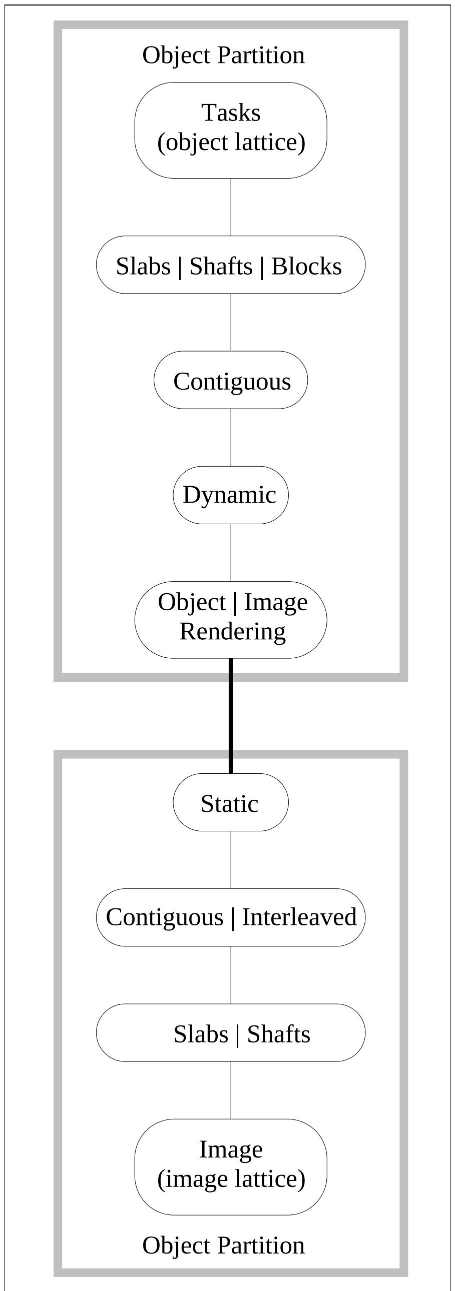


Fig. 6.9 - Object-partition options