Dynamic Load Balancing through query partitioning for the Web Map Services

GIS services and specifically Map Services are very crucial for the spatial decision making and situation assessment. In order for the decision makers to access data and make situation assessment seamlessly, maps should be created interactively in time and correctly. Maps are basically composed of complex comprehensible data created with spatial data sets which are mostly kept in Databases or in plain text files. GIS often contains spatial data represented as large sets of points, chains of line segments, and polygons. Since interactive Geo-Science applications require quick response times, the GIS services must enable accessing and processing these large data sets in a reasonable time period.

Load balancing is considered to improve the performance and scalability of the computer based systems. Load balancing techniques can be grouped into 2 categories. These are static and dynamic load balancing techniques. Static load balancing methods divide and allocate the data pair prior to the computation process. In contrast, dynamic load balancing techniques divide and allocate work at runtime. The both classes of the load balancing techniques have the same aims. These are increasing the scalability, high performance and, high availability and disaster recovery. *We will be using dynamic load balancing in our applications.*

There are three well-known classical dynamic load balancing algorithms to share the work among the worker nodes. These are round-robin (RR), least connection and weighted RR. RR directs the network connection to the next server, and treats all servers as equals, regardless of the number of connections or response times. Least Connection (LC) directs the network connection to the server with the fewest connections. Weighted RR allows you to assign performance weight to each server in which it is similar to RR and LC, however, servers with higher weighted value receive a larger percentage of connections at any one time. We will be using only RR, if we encounter a case in which we have more partition than the available servers. In all the other cases, general load balancing techniques do not work here *due to the nature of the spatial data (see the following section).*

Load balancing is based on the partition of the main work into smaller pieces. We do the partition on the queries to Map Server. Map Server behaves as failover and backup service by determining intelligently which server to assign the work to. Worker servers might be other cascaded WMSs for map images or Web Feature Services for the feature data depending on the clients' requests. Details are explained in the following sections.

Why do the classical load balancing algorithms not work? :

The short answer is that we don't know the workload previously. The work is partitioned into independent work pieces, and the work pieces are of highly variable sizes. It is not possible to estimate the size of total work at a given server. Partitioning of feature data (represented in sets of polygons, line

strings and points) is hard due to the spatial nature, varying sizes of the feature collections, and uncertainty about the query location.

Web Map Services are queried based on some criteria. Load balancing and the partitioning are based on the values of the "*bounding box*" criteria. Bounding box values define the location of the places whose map images are going to be created. Bounding box values are in the form of (minx, miny, maxx, maxy). For example, in Figure 1(a) client needs a map of earth defined by bbox value of (a,b,c,d).

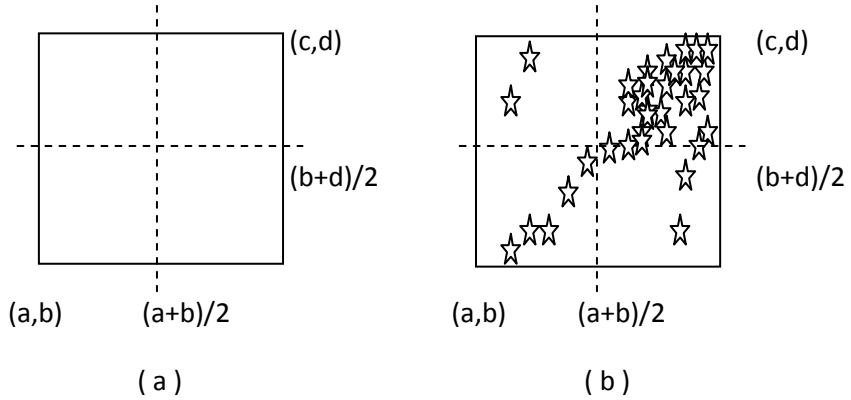


Figure 1: Classic partitioning can not share the work equally among worker servers. Server assigned the partition of "((a+b)/2, (b+d)/2), (c, d)" gets the most of the work.

Proposed dynamic load balancing algorithm:

Query partitioning for the variable sized and un-evenly distributed data

We think caching and dynamic load balancing all together in our proposed solution. Our main concern in load balancing is the determining the best partition of the application data. As it is shown in Figure 1, classical partition (half and half) does not help to share the work to the servers equally every time. In our proposed architecture we utilize from the caching and the most importantly *critical regions*.

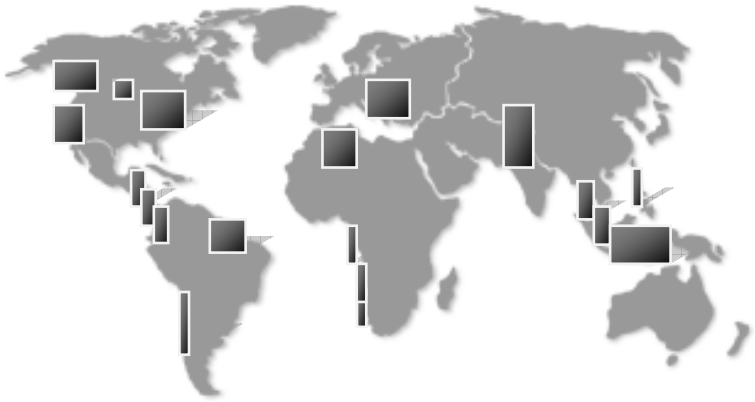


Figure 2: Globe showing predefined sample critical regions

Critical Regions: (see sample distribution of critical regions in Figure 2) Critical region is basically data intensive regions defined by a specific search criterion. In the GIS case critical search criteria is the bounding box values. Therefore, critical regions in GIS are defined as data intensive regions defined by set of boundary boxes. Initially we define it in the properties file of the Mapping Server (WMS-Extd) before run time manually. There are two ways to define the critical regions by this way. The easiest one is contacting the data provider's administrators and get from them. Another way is little more complicated such that WMS-Extd admin pre-runs the system and make a request to data provider with the widest range of the bbox value. After getting critical data from data provider, admin defines the critical regions in smaller bboxes by looking at the returned critical data plotted on image maps. In the future, we wilt try to figure out a protocol allowing this to come from data providers (which are WFS in our application framework) in a 2-dimm array or in a text file.

We group the requests into two, first time calls and successive calls. First time calls are done without caching on data. Successive calls utilize the caching. Therefore, dynamic load balancing algorithm will be changing depending on the request type.

1. Load Balancing for the first time calls (No cached data):

We have four possible sub cases to deal with. First is all the main request bbox falls in a critical region, second is some bbox fall in critical region, third is none of bbox falls in critical region and last is bbox covers critical region. In cases of first and third, we partition the bbox into four equal sizes as shown in Figure 1 (a) and (b) respectively. In case of second and last, we do a partition as in Figure 3. Dashed lines intersection represents the partition on main request. They cut the overlapped critical region into four equal sizes. It defines the partitions of the main request indirectly. In the Figure 5, CR represents "critical region bbox".

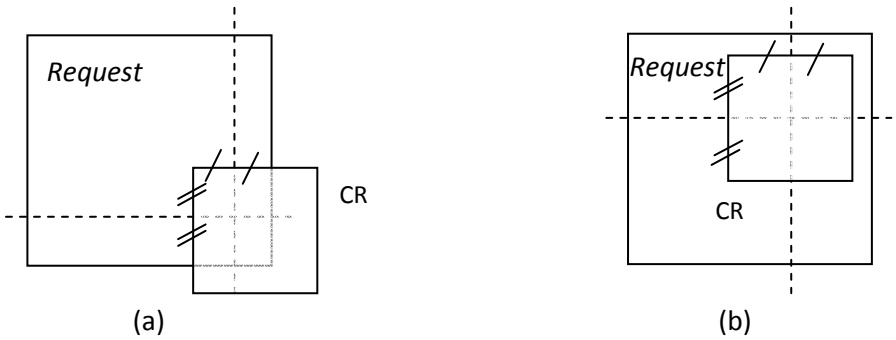
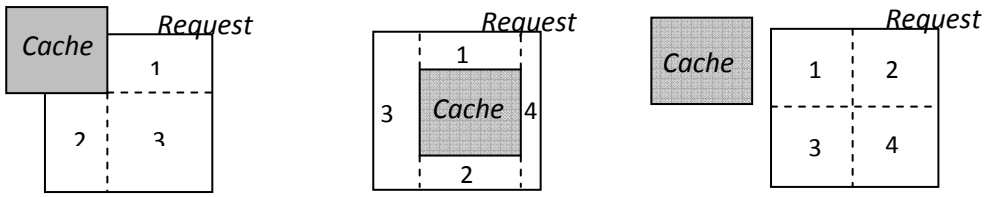


Figure 3: Partitioning in case of first time calls- no caching. (a) Some parts of request bbox fall in a critical region and (b) in case of request bbox covers critical region.

Furthermore, in case of Figure 3 (a) if the overlapped region is smaller than the one forth of the total CR or main request bbox has some or full portion of more than one CR then, make the partition as in Figure 1 (a) (basic load balancing).

2. Load balancing for the successive calls (utilize from the cached data):

We first utilize the caching, if caching is not enough to meet the requested data, we make bbox partitioning for the remaining regions in accordance with the load balancing algorithms. There are two generalized cases. In first case caching doesn't meet the requested data as shown in Figure 4. Figure 4 shows three possible cases of load balancing with caching. (a) represents partially overlapped region, (b) represents fully overlapped region and (c) represents the cases with no overlapped regions. In (c) application can not utilize from the load balancing. In the second case, (do nothing case shown in Figure 5), caching meets the requested data. In other words, successive request's bbox is inside the cached image's bbox



(a) (b) (c)

Figure 4: Possible partitioning cases for the load balancing with caching in successive calls

After removing the cached region from the requested bbox, we partition the remaining parts as much as equally (see Figure 4). The partitions obtained in Figure 4 look like the requests in the category of the first time calls whose algorithm is explained in Figure 3 and previous section.

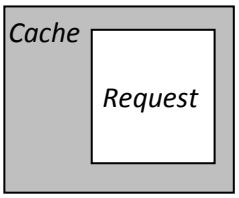


Figure 5: Do nothing case

In order for the proposed load balancing algorithms to work, we put the critical data on which we apply the algorithms on top of the other layers in the map images, and we make them transparent to see the other layers such as boundary lines, and global world-map coming from other servers.