

Inside LiquiFire:

LiquiFire Clustering Techniques

Inside LiquiFire: LiquiFire Clustering Techniques

Why Cluster

Like all machines, computers have limitations to their capacity to perform the work for which they are designed. In a manufacturing environment, for example a particular injection-molding machine may be able to produce a maximum number of parts per hour. Once this level of capacity is attained, to produce parts at a faster rate, not only must an additional injection-molding machine be added to the plant, but all the supporting processes must as well be duplicated. Beyond the costs of the additional injection-molding machine, additional material supply systems and parts collection systems must be purchased, and new scheduling tasks must be addressed within the workflow. And, although production can be increased through this first duplication, constraints in the supporting processes prevent similar gains from subsequent additional systems – efficiency can not be maintained.

Clusters are different. The purpose of a clustered environment is to provide additional capacity while removing the need to duplicate any supporting infrastructure. The result is an increase in capacity merely by the adding new nodes to the cluster, but without requiring or creating additional overhead. Whether a single node or a cluster, asset management does not change, and requests are sent in exactly the same fashion.

Clustering allows a group of servers to work in concert, while appearing to users or clients as a single, more powerful server. In a clustered environment, there should be no distinction between a single machine and a cluster, aside from the additional capacity that the clustered environment represents.

Clustering in LiquiFire

LiquiFire® provides facilities to enable the implementation of transparent clustering solutions. While many configurations of LiquiFire clusters can be easily architected, two common and representative solutions will be discussed in detail in this paper.

In order to understand how LiquiFire clusters can be architected, it is first important to understand how LiquiFire processes an imaging request.

Figure 1 depicts a single LiquiFire server. Here, requests are first checked against the **render cache** to see if a rendered image already

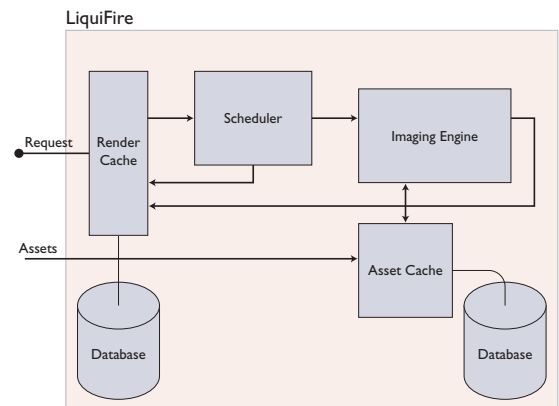


Figure 1: Single LiquiFire

Are your images fluid?™

exists for this request. If a match is found, the cached image is returned and processing is complete. If a cached image is not present, the request is analyzed and assets are identified. Assets located at external locations are fetched and cached; those stored locally or available from the **asset cache** are acquired for use. LiquiFire then performs the requested imaging operations via the **imaging engine**, caches the resulting image in the **render cache**, if appropriate, and returns the image to the client.

Master-Slave Clustering

Figure 2 depicts a two node Master-Slave cluster. In this architecture, one machine in the cluster is the **master** and the other the **slave**. The master node is responsible for scheduling requests among the available nodes (including the master itself), while the slaves in the cluster are solely responsible for performing imaging operations.

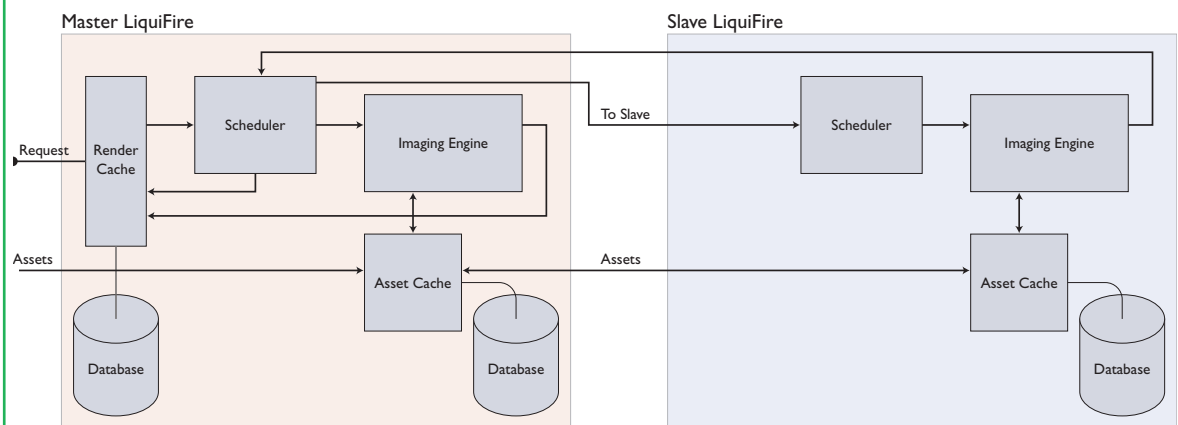


Figure 2: **Master-Slave Cluster**

In this model, requests are checked against the render cache in the Master, and if a cached result exists, this image is returned to the client. When a cached image is not available, the Master's scheduler, using heuristics as well as performance history, network congestion, and server-availability statistics, determines which member of the cluster is most appropriate to serve this request. The request is forwarded to the appropriate node (master or slave) and processing continues as in the single node architecture.

Assets required during the rendering process are identified, acquired and stored in the node-local asset cache. During quiescent periods, each node's asset cache is n-way synchronized with those in other members of the cluster, offering further efficiencies, and allowing both cached and locally stored assets on each node to be universally available.

Master-Master Clustering

Figure 3 illustrates a two node Master-Master cluster. This architecture presents a trade-off in cluster design. In this configuration both nodes perform as autonomous servers, requiring the addition of an external load-balancing mechanism to be added to the architecture. This addition presents an important new capability. In normal operating conditions, capacity is increased, yet in the event of failure by either one of the nodes, the cluster continues to perform uninterrupted. To implement this configuration, both nodes are designated as Masters, and each is also identified to the other

as a Slave. This twist enables the n-way asset synchronization to occur, while not allowing either master to off-load a request to a slave node.

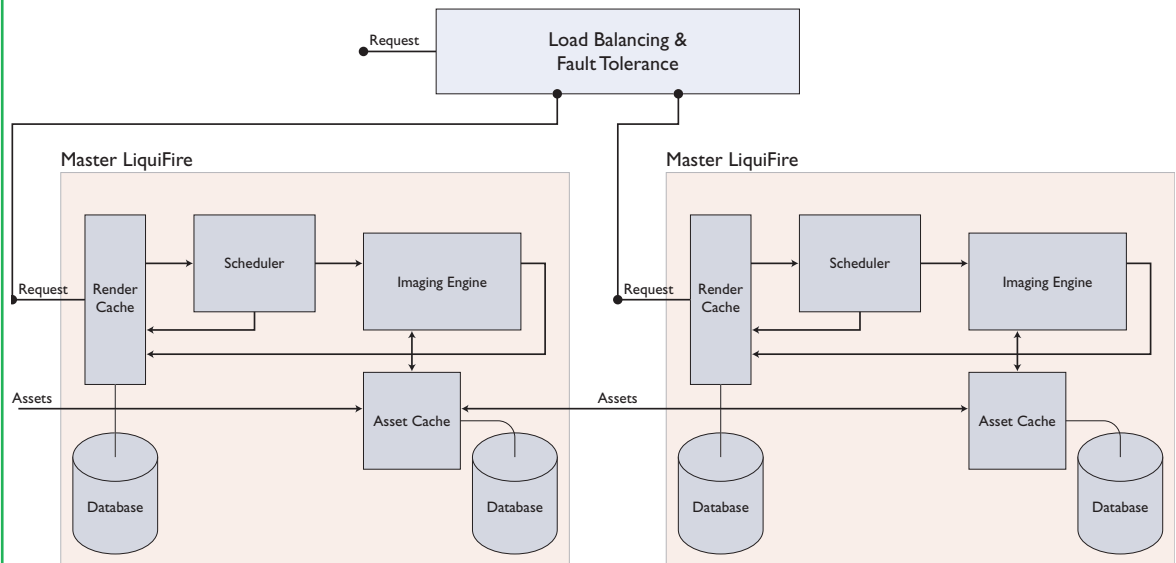


Figure 3: **Master-Master Cluster**

Overflow Protection

Consider a Master-Slave cluster where one of the nodes designated as a slave is, in fact, a whole cluster itself, located at LiquidPixels’ hosting facility. In this scenario, under normal conditions, requests are handled by the local Master node. But under sudden or severe levels of requests, the Master can decide to off-load requests by forwarding them to the LiquidPixels’ datacenter-based cluster, offering an environment capable of processing transient spikes in request volume without concern.

Conclusion

LiquiFire’s unique architecture allows for tremendous flexibility and versatility of cluster design. This enables LiquiFire processing clusters to be simply and robustly configured for any environment from fault tolerance to greater capacity. Because of LiquiFire’s unique architecture, asset management, network latency management, node failure, and node-to-node scheduling are fully automated – without the need for any external facilities.

About LiquidPixels

Founded in 2000, LiquidPixels, Inc. provides LiquiFire® — the premier middleware solution for enterprise dynamic imaging.

LiquiFire products yield greatly reduced costs, streamlined workflows, and easy integration while significantly enhancing online environments, increasing the likelihood of purchase, and uniquely fulfilling the one-to-one promise of the Internet. The patent-pending LiquiFire suite of solutions is available both as enterprise servers and as a hosted service.

A privately held company, LiquidPixels operates facilities in Rochester, NY and Boston, MA. For more information, visit www.liquidpixels.com, call 866.808.4937, or write to info@liquidpixels.com.