Load Balancing

Replication is an important IceGrid feature but, when combined with load balancing, replication becomes even more useful.

IceGrid nodes regularly report the system load of their hosts to the registry. The replica group's configuration determines whether the registry actually considers system load information while processing a locate request. Its configuration also specifies how many replicas to include in the registry's response.

IceGrid's load balancing capability assists the client in obtaining an initial set of endpoints for the purpose of establishing a connection. Once a client has established a connection, all subsequent requests on the proxy that initiated the connection are normally sent to the same server without further consultation with the registry. As a result, the registry's response to a locate request can only be viewed as a snapshot of the replicas at a particular moment. If system loads are important to the client, it must take steps to periodically contact the registry and update its endpoints.

On this page:

Replica Group Load Balancing

A replica group descriptor optionally contains a load balancing descriptor that determines how system loads are used in locate requests. The load balancing descriptor specifies the following information:

- Type
  Several load balancing types are supported.

- Sampling interval
  One of the load balancing types considers system load statistics, which are reported by each node at regular intervals. The replica group can specify a sampling interval of one, five, or fifteen minutes. Choosing a sampling interval requires balancing the need for up-to-date load information against the desire to minimize transient spikes. On Unix platforms, the node reports the system's load average for the selected interval, while on Windows the node reports the CPU utilization averaged over the interval.

- Number of replicas
  The replica group can instruct the registry to return the endpoints of one (the default) or more object adapters. If the specified number *N* is larger than one, the proxy returned in response to a locate request contains the endpoints of at most *N* object adapters. If *N* is 0, the proxy contains the endpoints of all the object adapters. The Ice run time in the client selects one of these endpoints at random when establishing a connection.

For example, the descriptor shown below uses adaptive load balancing to return the endpoints of the two least-loaded object adapters sampled with five-minute intervals:

XML

```
<replica-group id="ReplicatedAdapter">
    <load-balancing type="adaptive" load-sample="5" n-replicas="2"/>
</replica-group>
```

The type must be specified, but the remaining attributes are optional.

As of Ice 3.5, IceGrid ignores the object adapters of a disabled server when executing a locate request, meaning the client that initiated the locate request will not receive the endpoints for any of these object adapters.

Load Balancing Types

A replica group can select one of the following load balancing types:

- Random
  Random load balancing selects the requested number of object adapters at random. The registry does not consider system load for a replica group with this type.

- Adaptive
  Adaptive load balancing uses system load information to choose the least-loaded object adapters over the requested sampling interval. This is the only load balancing type that uses sampling intervals.

- Round Robin
  Round robin load balancing returns the least recently used object adapters. The registry does not consider system load for a replica group with this type. Note that the round-robin information is not shared between registry replicas; each replica maintains its own notion of the "least recently used" object adapters.

- Ordered
  Ordered load balancing selects the requested number of object adapters by priority. A priority can be set for each object adapter member of the replica group. If you define several object adapters with the same priority, IceGrid will order these object adapters according to their order of appearance in the descriptor.

Choosing the proper type of load balancing is highly dependent on the needs of client applications. Achieving the desired load balancing and fail-over behavior may also require the cooperation of your clients. To that end, it is very important that you understand how and when the Ice run time uses a l ocator to resolve indirect proxies.

Using Load Balancing in the Ripper Application

The only change we need to make to the ripper application is the addition of a load balancing descriptor:

XML

```
<icegrid>
    <application name="Ripper">
        <replica-group id="EncoderAdapters">
            <load-balancing type="adaptive"/>
            <object identity="EncoderFactory" type="::Ripper::MP3EncoderFactory"/>
        </replica-group>
        <server-template id="EncoderServerTemplate">
            <parameter name="index"/>
            <parameter name="exepath" default="/opt/ripper/bin/server"/>
            <server id="EncoderServer${index}" exe="${exepath}" activation="on-demand">
                <adapter name="EncoderAdapter" replica-group="EncoderAdapters"
                    endpoints="tcp"/>
            </server>
        </server-template>
        <node name="Node1">
            <server-instance template="EncoderServerTemplate" index="1"/>
        </node>
        <node name="Node2">
            <server-instance template="EncoderServerTemplate" index="2"/>
        </node>
    </application>
</icegrid>
```

Using adaptive load balancing, we have regained the functionality we forfeited when we introduced replica groups. Namely, we now select the object adapter on the least-loaded node, and no changes are necessary in the client.

Interacting with Object Replicas

In some applications you may have a need for interacting directly with the replicas of an object. For example, the application may want to implement a custom load-balancing strategy. In this situation you might be tempted to call `ice_getEndpoints` on the proxy of a replicated object in an effort to obtain the endpoints of all replicas, but that is not the correct solution because the proxy is indirect and therefore contains no endpoints. The proper approach is to query well-known objects using the `findAllReplicas` operation.

See Also

- Object Adapter Replication
- Connection Establishment
- Replica-Group Descriptor Element