# Background Tasks

Each Yellowfin node will be configured by default to run background tasks. The cluster should be properly configured for task execution, depending on the cluster type.

## LEGACY Cluster

Apply the following configurations if using a LEGACY cluster.

### Task Scheduler

Background tasks should be enabled on only one node. This is because tasks such as report broadcast could potentially send a report multiple times to end users, when multiple nodes run the same task. Because of this it is recommended to limit background tasks to a single node.

Disable the task scheduler by adding the following to the web.xml file, inside the MIStartup Servlet block.

```
<init-param>
        <param-name>DisableTaskScheduler</param-name>
        <param-value>TRUE</param-value>
</init-param>
```

### System Tasks

System tasks should be enabled only on one node.

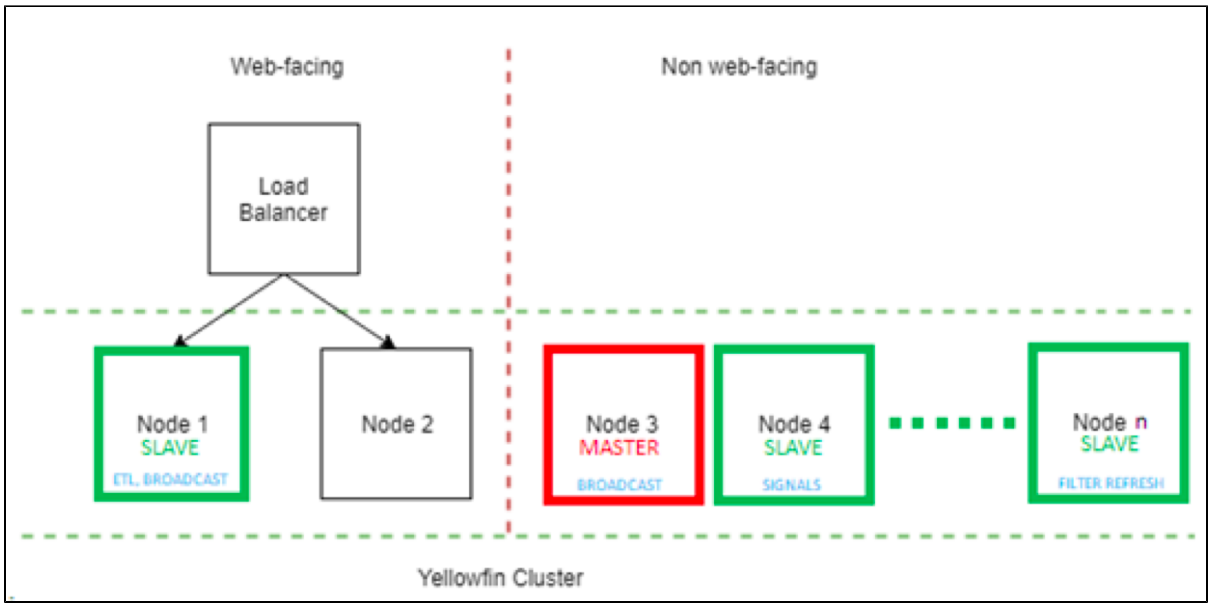
Disable the background system tasks by removing (or commenting out) the following XML block from web.xml. This will disable system tasks, such as, Group Flattening, LDAP synchronization, Event Archiving, Document Cleanup, and Average Run time calculations.

```
<servlet>
      <servlet-name>SystemTaskManager</servlet-name>
      <servlet-class>com.hof.servlet.SystemTaskManager</servlet-class>
      <load-on-startup>8</load-on-startup>
</servlet>
```

---

## DYNAMIC/REPOSITORY Cluster

A DYNAMIC or REPOSITORY cluster is highly scalable for background execution. To enable this, all nodes should have **AutoTaskDelegation** set to TRUE. The Yellowfin cluster can then operate in the configuration described below.




The Master node coordinates task delegation. It may optionally run background tasks. The Master always runs System Tasks, unless explicitly disabled (not recommended). A Slave node runs only the background tasks which it is configured to run.

The Yellowfin cluster is highly configurable.

- A node may be configured to become Master but never a Slave. To do this, do not include any TaskType in the ClusterManagement servlet block in web.xml. If multiple nodes are configured to become the Master to allow for failover, then the node that gets started first becomes the Master.
- A node may be configured to become a Slave but never a Master. To do this, add the following to the MIStartup servlet block in web.xml:

```
<init-param>
        <param-name>DisableTaskSchedulerPolling</param-name>
        <param-value>TRUE</param-value>
</init-param>
```

- Web-facing nodes may be configured to never become a Master or Slave. This way, background tasks don't compete for resources with online processing. To do this, add the following to the MIStartup Servlet block in web.xml:

```
<init-param>
        <param-name>DisableTaskScheduler</param-name>
        <param-value>TRUE</param-value>
</init-param>
```

- Nodes may be configured to run specific tasks. The maximum number of concurrent tasks of each type should also be configured. To do this, add the following snippet in the ClusterManagement servlet block in web.xml.

```
<init-param>
              <param-name>TaskTypes</param-name>
              <param-value>
                     FILTER_CACHE,
                     ETL_PROCESS_TASK,
                     SOURCE_FILTER_REFRESH
                             </param-value
</init-param>
<init-param>
              <param-name>MaxParallelTaskCounts</param-name>
              <param-value>
                     1,
                     5,
                     3
              </param-value>
</init-param>
```

This configures the node to run a maximum of one Filter Cache refresh task, five Data Transformation tasks and three Source Filter Refresh tasks at the same time.

If this snippet is not configured in the ClusterManagement servlet, the node will not run any background task.

## Task Types

Available task types and their descriptions:

| Task name | Description |
|---|---|
| REPORT_BROADCAST_BROADCASTTASK | Report Broadcast |
| REPORT_BROADCAST_MIREPORTTASK | Report data refresh |
| ETL_PROCESS_TASK | Data Transformation |
| FILTER_CACHE | Cached filter refresh |
| SOURCE_FILTER_REFRESH | Access filter refresh |
| SOURCE_FILTER_UPDATE_REMINDER | Access filter update reminder |
| ORGREF_CODE_REFRESH | Refresh org ref codes |
| THIRD_PARTY_AUTORUN | Third-party connector data cache refresh |
| SIGNALS_DCR_TASK | Pre-run task for Signals |
| SIGNALS_ANALYSIS_TASK | Signals Analysis |
| SIGNALS_CLEANUP_TASK | Post-run task for Signals |
| COMPOSITE_VIEW_REFRESH | Refresh data for composite views |
| SIGNALS_CORRELATION_TASK | Detecting correlations in Signals |

Here are some recommendations:

- Determine which tasks will be run repeatedly in the system.
- Dedicate nodes for high frequency, resource intensive tasks, such as Signals and Data Transformations.
- A node can run as many tasks in parallel, as the number of cores. However, some tasks may have a high ratio of wait time to compute time, which means it can still run tasks while others are waiting. One would have to profile the application to fine tune the total number of threads. Yellowfin recommends not more than five threads per core. This number may then be divided amongst the tasks which are configured to run on the node. For example, if the deployment runs many Cached Filter refresh tasks, a higher number should be set as its “MaxParallelTaskCount”.

- The number of threads spawned for all background tasks may be controlled using the following parameters:
    - **TaskSchedulerThreads** – maximum number of threads available for all background tasks
    - **TaskSchedulerMaxThreadQueue** – number of slots in the “wait” queue. Items in the wait queue are run as soon as a thread becomes available, without having to wait for the next cycle.

Consider the following configuration:
TaskSchedulerThreads = 5
TaskSchedulerMaxThreadQueue = 15
TaskType | MaxParallelTaskCount = FILTER_CACHE | 8
ETL_PROCESS_TASK | 3
SIGNALS_ANALYSIS_TASK | 10

The node can run 5 concurrent tasks of any of the three configured types.

When 5 tasks are running, the next task will be added to the “wait” queue. As soon as a task finishes execution, the waiting task will begin running.

When 5 tasks are running and 15 tasks are waiting for a free thread, the next task will be rejected.

Ideally, (TaskSchedulerThreads + TaskSchedulerMaxThreadQueue) = MaxParallelTaskCount

## Signals and Clustering

A node running only SIGNALS_DCR_TASK, SIGNALS_ANALYSIS_TASK and SIGNALS_CLEANUP_TASK will be designated as a “Signals Node”. The node may run one or more of these types, but it shouldn’t run any others. A cluster may have one or more Signals Nodes. Signals nodes are specially licensed and cannot be used for processing web requests or other background tasks. A Signals node cannot become a Master.

```
<init-param>
              <param-name>TaskTypes</param-name>
              <param-value>
                     SIGNALS_DCR_TASK,
                     SIGNALS_ANALYSIS_TASK,
                     SIGNALS_CLEANUP_TASK
                            </param-value
</init-param>
<init-param>
              <param-name>MaxParallelTaskCounts</param-name>
              <param-value>2,5,2
              </param-value>
</init-param>
```

**Previous topic:** Application messaging
**Next topic:** Session replication