Dynamic Service User Guide

Platform Symphony
Version 5.1
April 2011

Copyright

We'd like to hear from you

You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this document, please address your comments to doc@platform.com.

Your comments should pertain only to Platform documentation. For product support, contact support@platform.com.

Document redistribution and translation

Internal redistribution

Trademarks

Third-party license agreements

http://www.platform.com/Company/third.part.license.htm

Third-party copyright notices

http://www.platform.com/Company/Third.Party.Copyright.htm

Contents

Dynamic Service

Dynamic Service is a feature of the Multi-core Optimizer add-on product. The Multi-core Optimizer reduces capital and operating expenses by improving utilization of resources in mixed environments. It can also reduce I/O and memory contention in multi-core environments. The Dynamic Service feature allows dynamic control of service-to-slot mapping to optimize core, memory, and I/O use. This enables efficient running of workload made up of multi-threaded, single-threaded, I/O-intensive, and compute-intensive tasks from a single application and/or a mix of applications in a multi-core/heterogeneous environment.

An additional software license is required to use this feature. This feature is packaged with Symphony and does not require separate deployment.

Scope

Applicability	Details
Operating system	• Windows • Linux • Solaris
Limitations	N/A

Feature licensing

Follow these steps when you want to add or update a license for the dynamic service feature.

1. Acquire a dynamic service license key from Platform.
2. Append the license key to $EGO_CONFDIR/license.dat. Note that all management hosts running an SSM must have access to this license file.
3. Configure the dynamic service feature.
4. Register and enable the new application profile.

Note:

If your application was already configured and registered before you acquired the license, you need to disable and re-enable the application so that the license is verified and the feature is enabled.

About slot usage requirements

Some workload within the same application may be more compute-intensive than others. The dynamic service feature allows you to run each type of workload on resources according to its own resource requirements. In this manner, the workload consumes slots according to its own slot usage requirement.

SSM resource scheduling

The SSM assigns idle resources to sessions in the following manner:

- The SSM calculates each session's deserved share according to its scheduling policy, i.e., proportional, minimum services, etc.
- Every scheduling cycle, the scheduling policy tries to assign any idle resources to under-allocated sessions in accordance with proportion, as best it can, where:

  1. The resources taken must match the session's slot usage requirements.
  2. The resources taken must also be the best fit from among the remaining idle resources. For example, if the SSM is scheduling a session with a 1-slot requirement, it will try to choose 1-slot hosts first, then 2-slot hosts, and so on.
- If the under-allocated sessions cannot make use of all of the resources (for example, the remaining resources do not match the slot usage requirements for the under-allocated sessions), the policy tries to assign the resources to sessions that have already met or exceeded their deserved share. This means that the resources do not remain idle.

Resource group filtering

If resource groups are specified at the application level, it means that any sessions serving the application can be allocated resources from the specified resource groups. When a session with multi-slot usage requirements needs resources, it is possible that there are only hosts with an insufficient number of slots available since the required multi-slot hosts may have already been allocated to single-slot sessions. The multi-slot session would have to wait until suitable resources were available. If resource groups could be specified at the session level, you could request multi-slot hosts for the multi-slot sessions and, at the same time, prevent single-slot sessions from taking multi-slots hosts. The resource group filter can be used for this purpose. Each session can use a resource group filter to narrow down the choices of resource groups specified at the application level.

When clients submit workload for sessions with unique slot usage and resource group filter requirements, the SSM makes an allocation request to EGO based on the specified requirements. EGO goes through the resource groups and tries to satisfy the demand. (The order in which EGO goes through the resource groups is defined by the order that the resource groups appear in the resource plan in the `ConsumerTrees.xml` file.) After receiving resources from EGO, the SSM assigns resources to its sessions according to the configured policy, the slot usage requirement for each session, and the resource group filter for each session.

Workload preemption

If a session requires multiple slots to run its tasks and there are not enough idle resources to satisfy the session, the session must wait. In a busy cluster, this session may starve if other sessions occupy the slots. This situation may also apply to partial-slot tasks that are waiting to get slots from tasks with even smaller slot requirements. To resolve this problem, you can employ session-level preemption.

You can configure a session to be preemptive so that when the session is under-allocated, it can preempt workload of other sessions instead of waiting for other sessions to voluntarily release slots. For example, when a single-slot session finishes, the multi-slot session cannot immediately use the slot. In the meantime, another session may take the slot causing the multi-slot session to wait for other resources.

A session can be defined as preemptive or non-preemptive, the default being non-preemptive. A non-preemptive session does not trigger preemption and waits until the next resource is freed up. For example, if a 4-slot task from a non-preemptive session is to run, it must wait for 4 slots on the same host to free up at the same time.

The following criteria is used to identify preemption candidates in the given order:

1. Select all over-allocated sessions.
2. Order preemption candidates by preemption rank and choose the sessions with lowest rank. If there are multiple sessions in the lowest rank, choose the session according to the preemptionCriteria configuration, i.e., either a session with the most recently started task or the most over-allocated session (default).

   In cases where a few tasks must be preempted at the same time, the tasks with the smallest sum of elapsed computation times are chosen.

Session preemption characteristics:

- Preemption can only be triggered by under-allocated sessions.
- An under-allocated session that is configured to be preemptive can preempt over-allocated sessions if the under-allocated session has any unsatisfied demand.
- Workload preemption only happens if the under-allocated session's rank is higher than or equal to the rank of the over-allocated session. Whether a session preempts another session of lower or equal rank or only preempts a session of lower rank is determined by the setting of the preemptionScope attribute in the application profile. Sessions cannot preempt other sessions with higher rank.
- Only over-allocated slots will be preempted from a session. Preemption will not cause any session to become under-allocated.
- If multiple slots are shared by more than one session, the session with the highest session rank is taken into consideration. In this case, the lower rank session is "protected" by the higher rank session and will not be preempted. Similarly, if multiple sessions on one host share the same slot, the session with the highest preemption rank is taken into consideration.
- If multiple slots on the same host are shared by more than one session where some slots belong to an over-allocated session and some slots belong to an under-allocated session, these slots will not be preempted. In this case, the over-allocated session is "protected" by the under-allocated session. Similarly, if multiple sessions on one host share the same slot where one session is over-allocated and ther other is under-allocated, the slot will not be preempted.
- If the SSM cannot find any slots in the lowest rank sessions to preempt, either because these slots cannot be used by the session or these slots are "protected" by higher ranked sessions or under-allocated sessions, the SSM will consider the next higher ranked sessions.
- Preemption takes effect immediately. The service instance manager and service instance are restarted and assigned to under-allocated sessions.

Exclusive allocation

One of the main benefits of the dynamic service feature is to enable sessions with multi-core requirements to run workload. In a cluster made up of multi-core hosts, it is possible that, over time, a host’s usage can become fragmented. This can happen when a single-thread task is allocated a multi-core host. In this case, the task occupies one core but the remaining cores are not used. This problem can extend to many hosts in the cluster to the point where there are a lot of free slots but there are no hosts that are totally free.

Here is an example:

Suppose we have two 8-core hosts and two applications: one with 1-core sessions and one with 4-core sessions. If there are five 1-core tasks from App1 occupying 5 of 8 slots on HostA and then a 4-core task is submitted by App2, then it may get the remaining 3 slots on HostA and another 1 slot on HostB. However, this configuration of slots is not usable by the 4-core task in App2.

Exclusive host allocation, which is configured through the Platform Management Console for each resource group in the resource plan, can be used to resolve the resource fragmentation problem.

When Exclusive is set, all the slots of each host in the resource group are assigned to only one consumer at a time. So if there is one slot on a host allocated to one consumer, remaining slots on the same host will not be allocated to other consumers; these slots will only be allocated to this consumer when it has demand. When a consumer reclaims resources from another consumer, EGO enforces the reclaiming of all slots on a host. Note that this behavior may cause a consumer to be allocated less slots than it deserves since a consumer can only be allocated slots on a host when it deserves the whole host. The same principle applies to a consumer that wants to reclaim slots.

When Exclusive is set, if the number of slots on a host is increased, EGO does not allocate the extra slots to the application until existing allocations on the host are released.

Configuring preemption, preemption rank, service-to-slot ratio (slot usage), and resource group filter

Preemption, service-to-slot ratio, resource group filter, and preemption rank are all configured in the SessionType > Type element of the application profile.

Preemption scope is configured in the Consumer element of the application profile. The default value for preemptionScope is LowerOrEqualRankedSessions.

If preemptive is set to true for a session type, the under-allocated sessions of this session type can preempt other over-allocated sessions. The default value for preemptive is false.

The preemptionRank attribute defines the session’s rank in relation to other sessions. (Sessions with a lower preemption rank will get preempted before sessions with a higher rank.) The default value for preemptionRank is 1.

The serviceToSlotRatio attribute defines the number of slots required to run a service instance. The default ratio is 1:1.

The resourceGroupFilter is a list of resource groups from which the session can use resources. The default value is empty, which means resource groups specified at the application level will be used.

Here is an example of the attributes configuration in the application profile:

```
<Consumer preemptionScope="LowerRankedSessions"/>
<SessionTypes><Type name="type1" … serviceToSlotRatio="1:4" resourceGroupFilter="RG1
RG2" preemptive="true" preemptionRank="2"/>
```

Overriding configured parameters via API

At session creation time, a client can override the session type’s serviceToSlotRatio, resourceGroupFilter, preemptive, or preemptionRank via the API. Once the session is created, these parameters cannot be changed. Refer to the API Reference in the Knowledge Center for more information.

Best practices

Slot usage requirements

These guidelines apply when one slot is configured to equal one core.

Set the serviceToSlotRatio to 1:N for multi-threaded, CPU-intensive workload, where N = the number of threads.

Set the serviceToSlotRatio to N:1 for lightweight services such as services that perform a lot of I/O.

Resource groups and filters

Organize hosts into resource groups according to the number of cores on each host and create a resource plan for each resource group. For example, all 1-core hosts should be in one resource group, all 4-core hosts should be in another resource group, etc.

For serviceToSlotRatio of 1:N, only specify resource groups with >= N-core hosts. For serviceToSlotRatio of N:1, specify any resource groups.

Preemption

Configure sessions with high slot requirements to be preemptive and to have higher preemption ranks; this enables those sessions to preempt sessions with lower slot requirements to avoid starvation. This also prevents sessions with equal slot requirements from preempting each another.

Here are some additional preemption guidelines for other types of workload. Note that the preemption rank values are provided as an example.

Type of Workload	Preemption Rank
For workload that you do not want preempted, set the preemption rank to the highest level.	20
For normal workload that can be preempted without consequence, set the preemption rank to the lowest level.	10

Exclusive allocation

1. Configure Exclusive for resource groups made up of multi-core hosts to prevent fragmentation of host slots.
2. Do not set Exclusive for resource groups containing management hosts. In one case, setting Exclusive may cause all the SSMs on a host to be restarted when an application associated with one of the SSMs is disabled. Similarly, stopping a system service on a management host running other system services, may cause the other services on that host to be restarted.
3. Do not configure Exclusive if the resource group contains heterogeneous hosts, especially if the ownership policy is configured. In such cases, it is difficult to configure consumer ownership properly to ensure the consumer can always get a host, since the consumer cannot get a host if the number of slots in the host is more than the consumer owns. For example, if there are two hosts left in the resource group (host1 has 2 slots and host2 has 4 slots) and consumer A owns 4 slots. Suppose consumer A has a demand for two slots and gets host1 first. Then, even if consumer A has more demand it cannot use host2 because its ownership does not allow it to use both host1 and host2.
4. Configure ownership to be a multiple of the number of slots per host in a homogeneous resource group.
5. Do not configure the exclusive slot policy in conjunction with the rusage feature that limits the number of service instances an application can run on a host; refer to *Limit the number of service instances that can run on a host* in the *Cluster and Application Management Guide*.

Index