
# Estimate Performance and Capacity Requirements for Workflow in SharePoint Server 2010

# Estimate Performance and Capacity Requirements for Workflow in SharePoint Foundation 2010 and SharePoint Server 2010

Eilene Hao Klaka
Mahesh Balasubramanian

Microsoft Corporation
June 2010

# Contents

This performance and capacity planning document provides guidance on the footprint that usage of Workflow has on topologies running SharePoint Server 2010.
For general information about how to plan and run your capacity planning for SharePoint Server 2010, see http://technet.microsoft.com/en-us/library/cc262971(Office.14).aspx.

# Test farm characteristic

## Hardware

Topologies used for these tests use a single back end computer for the content database and from one to four front-end computers with the default installation for Microsoft® SharePoint® Server 2010. Although the workflows used in these tests are not available in Microsoft SharePoint Foundation 2010, the results can be used to estimate similar scenarios on those deployments. The dataset used for these tests contains a single site collection with a single site based on the Team Site template on a single content database.

## Workload

Testing for this scenario was designed to help develop estimates of how different farm configurations respond to changes to the following variables:

- Impact of number of front-ends on throughput for manually starting declarative workflows across multiple front-end computers
- Impact of number of front-ends on throughput for automatically starting declarative workflows on item creation across multiple front-end computers
- Impact of number of front-ends on throughput for completing tasks across multiple front-end computers

It is important to note that the specific capacity and performance figures presented in this article will be different from the figures in real-world environments. The figures presented are intended to provide a starting point for the design of an appropriately scaled environment. After you have completed your initial system design, test the configuration to determine whether your system will support the factors in your environment.

## Test definitions

This section defines the test scenarios and provides an overview of the test process that was used for each scenario. Detailed information such as test results and specific parameters are given in each of the test results sections later in this article.

| Test name | Test description |
|---|---|
| **Throughput for starting workflows manually** | 1. Associate the out-of-box MOSS Approval workflow with a list that creates one task<br>2. Populate the list with list items<br>3. Call the StartWorkflow Web service method on Workflow.asmx against the items in the list for five minutes<br>4. Calculate throughput by looking at the number of workflows in progress |
| **Throughput for starting workflows automatically when an item is created** | 1. Associate the out-of-box MOSS Approval workflow with a list that creates one task, set to automatically start when an item is created |

| | |
|---|---|
| | 2. Create items in the list for five minutes<br>3. Calculate throughput by looking at the number of workflows in progress |
| **Throughput for completing workflow tasks** | 1. Associate the out-of-box MOSS Approval workflow with a list that creates one task, set to automatically start when an item is created<br>2. Create items in the list<br>3. Call the AlterToDo Web service method on Workflows.asmx against the items in the task list created by the workflows that started.<br>4. Calculate throughput by looking at the number of workflows completed |

# Hardware setting and topology

## Lab hardware

To provide a high level of test-result detail, several farm configurations were used for testing. Farm configurations ranged from one to four Web servers and a single database server computer that is running Microsoft SQL Server® 2008 database software. Testing was performed with one client computer. All Web server computers and the database server were 64-bit, and the client computers were 32-bit.

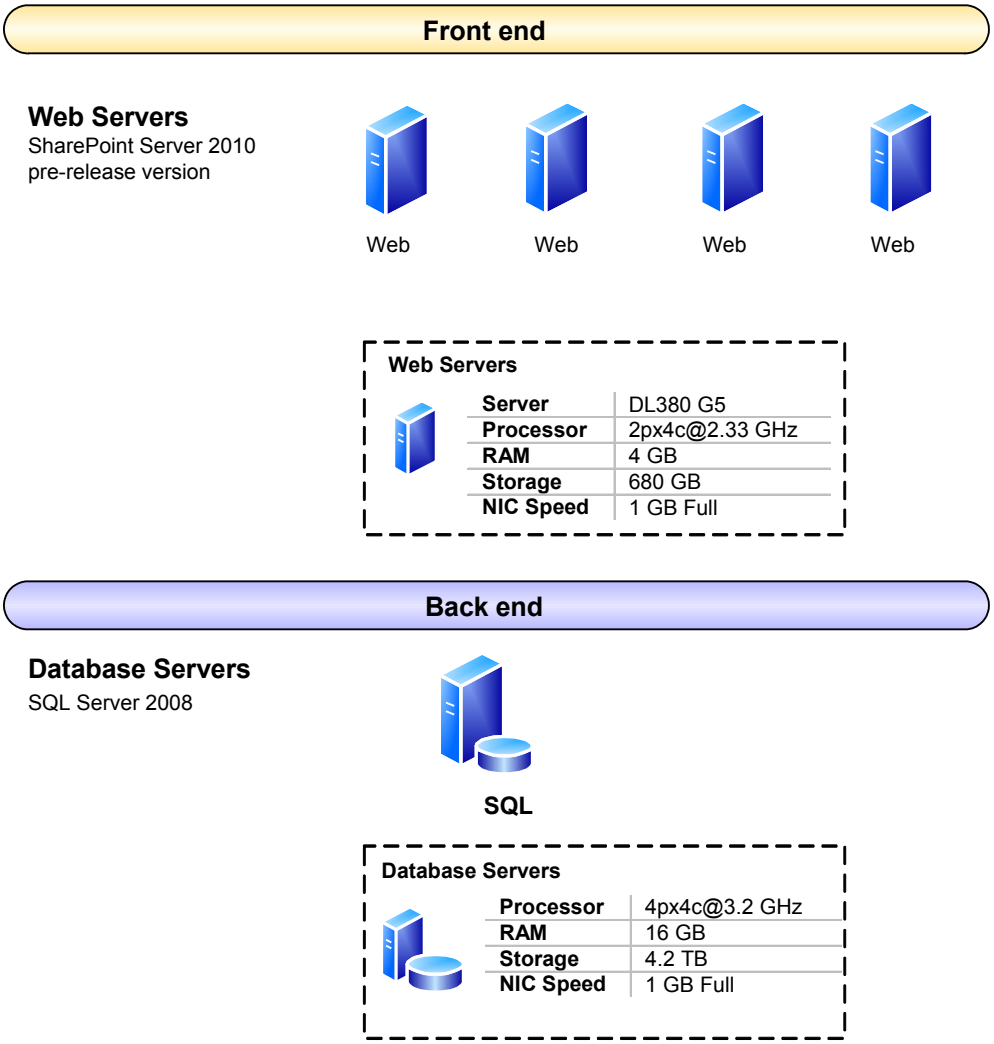
The following table lists the specific hardware that was used for testing.

| **Computer name** | **WFE1-4** | **SPSQL01** |
|---|---|---|
| **Role** | Front-end Web server | SQL Server back end (one computer) |
| **Processor(s)** | 2px4c@2.33GHz | 4px4c@2.4GHz |
| **RAM** | 4 GB | 16 GB |
| **Operating System** | Windows Server® 2008 R2 x64 | Windows Server 2008 R2 x64 |
| **Storage** | 680 GB | 4.2 terabyte |
| **# of NICs** | 2 | 2 |
| **NIC speed** | 1 gigabit | 1 gigabit |
| **Authentication** | NTLM | NTLM |
| **Software version** | 4747 | SQL Server 2008 R1 |
| **# of SQL Server Instances** | 1 | 1 |

| | | |
|---|---|---|
| **Load balancer type** | NLB | |
| **ULS Logging level** | Medium | Medium |

# Topology

### Workflow Test Farm Topology




# Dataset

To get benchmarks, most tests ran on a default Team Site on a single site collection in the farm. The Manual Start tests used a list with 8000 items to start workflows on.

# Test results

The following tables show the test results for Workflow in SharePoint Server 2010. For each group of tests, only certain specific variables are changed to show the progressive impact on farm performance.

Note that all the tests reported on in this article were conducted without think time, a natural delay between consecutive operations. In a real-world environment, each operation is followed by a delay as the user

performs the next step in the task. By contrast, in this testing, each operation was immediately followed by the next operation, which resulted in a continual load on the farm. This load can cause database contention and other factors that can adversely affect performance.

# Effect of front-end Web server scale on throughput

The following throughput tests were run using the out-of-box Approval workflow that ships with SharePoint Server 2010. The workflow association assigns one task, and all instances are run on a single list. Each instance of this workflow creates the following in the content database:

- An entry in the Workflows table to store workflow state
- Five secondary list items (1 task + 4 history items)
- Four event receivers to handle events on the workflow's parent item and task

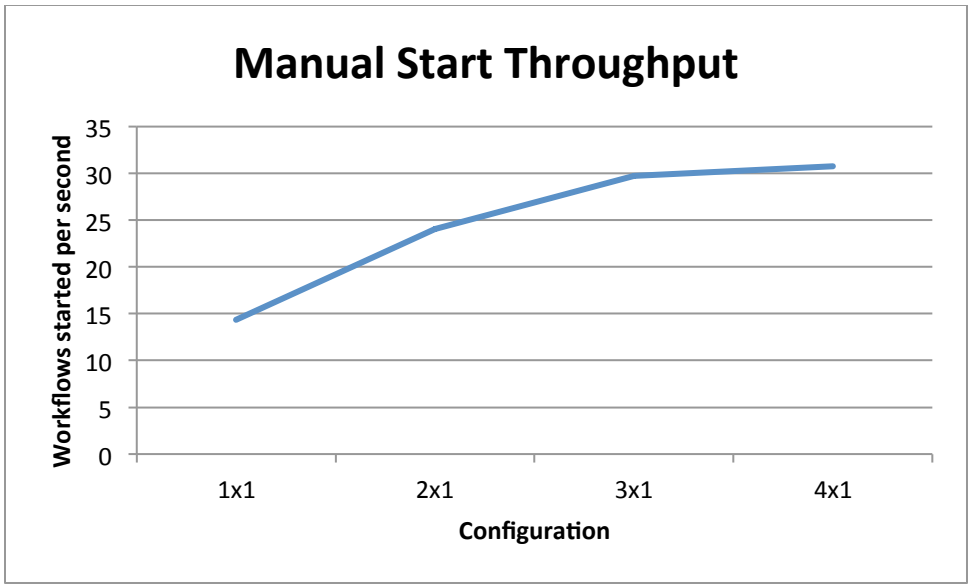
Workflow Postpone Threshold was set to be very large so that workflow operations would never get queued. Each test was done with five test runs for each test running for five minutes each.

## Manual start throughput

The test in the following table shows how the addition of front-end computers affects the throughput of starting workflows synchronously through the Web service. This test was run with a user load of 25 concurrent users continuously calling the StartWorkflow method on workflow.asmx and no other load on the farm. Note that the user load was the optimal load before getting dropped Web requests. The list is prepopulated with up to 8000 items.

| Topology | Approval Workflow Max RPS |
|---|---|
| **1x1** | 14.35 |
| **2x1** | 24.08 |
| **3x1** | 29.7 |
| **4x1** | 30.77 |

The following graph shows how throughput changes. Note that the addition of front-ends does not necessarily affect farm throughput in a linear manner but rather peaks off at around three to four front ends. In summary, the maximum throughput for manually starting workflows is around 30 workflows per second, and adding more than four front-ends will likely have a negligible impact.

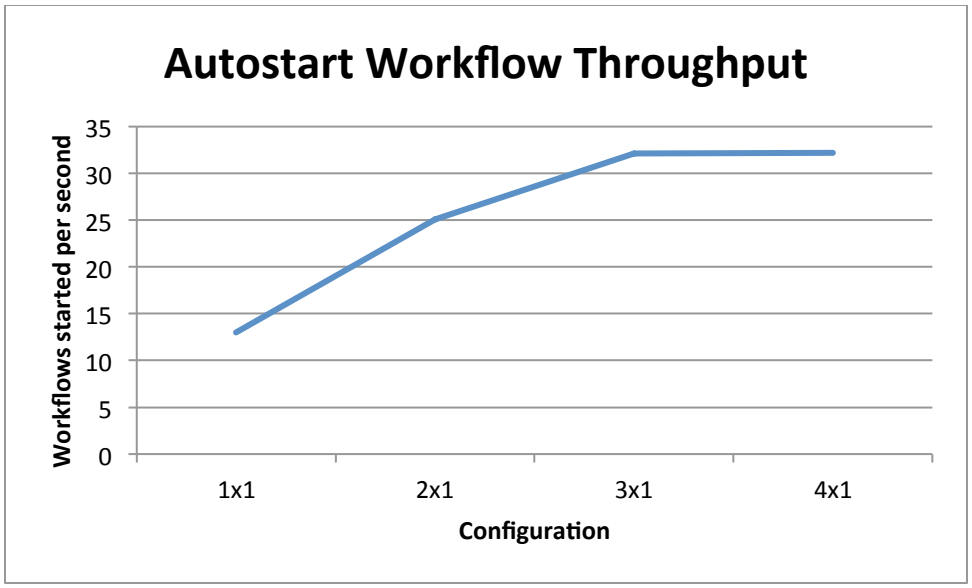



## Automatically starting workflows when items are created throughput

The test in the following table shows how the addition of front-end computers affects the throughput of starting workflows automatically when items are created. This test was run with a user load of 150 concurrent users continuously calling the list Web service to create new list items in single SharePoint list and no other operations on the server. The list started as an empty list.

| Topology | Approval Workflow Max RPS |
|---|---|
| **1x1** | 13.0 |
| **2x1** | 25.11 |
| **3x1** | 32.11 |
| **4x1** | 32.18 |

The following graph shows how throughput changes. The throughput is very close to the manual start operations. Similar to the manual start test, throughput peaks at around three to four front-ends at around 32 workflows per second maximum. Adding more than three or four front-ends will have a negligible effect.

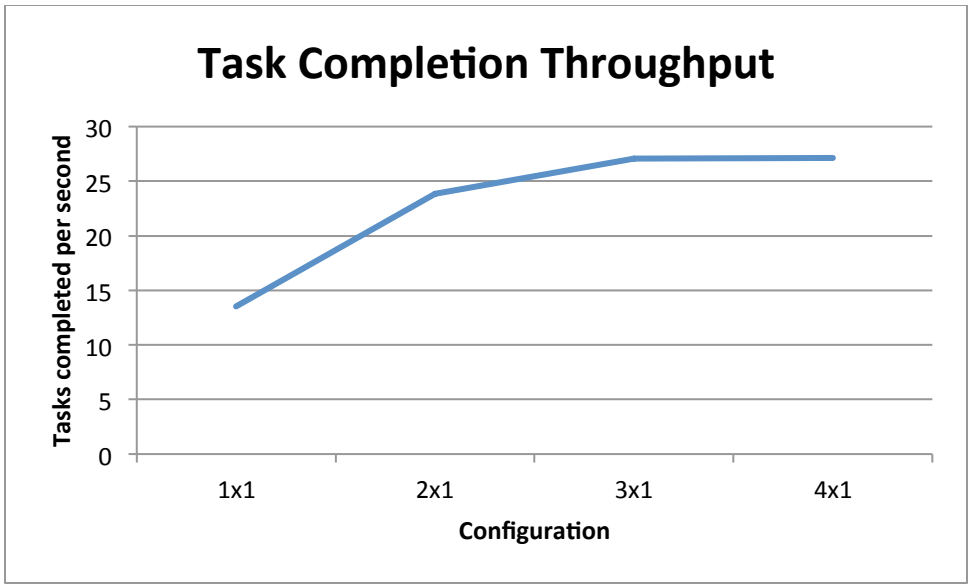



## Task completion throughput

The test in the following table shows how the addition of front-end computers affects the throughput of completing workflow tasks. The list used to complete tasks was the task list used by auto-start workflows in the previous test. This test was run with a user load of 25 concurrent users continuously calling the AlterToDo method on workflow.asm and no other operations on the server. The list started as an empty list.

| Topology | Approval Workflow Max RPS |
|---|---|
| 1x1 | 13.5 |
| 2x1 | 23.86 |
| 3x1 | 27.06 |
| 4x1 | 27.14 |

The following graph shows how throughput changes. Similar to the manual start test, throughput peaks at around 3-4 front-ends at around 32 workflows per second maximum. Adding more than three front-ends will have a negligible effect.

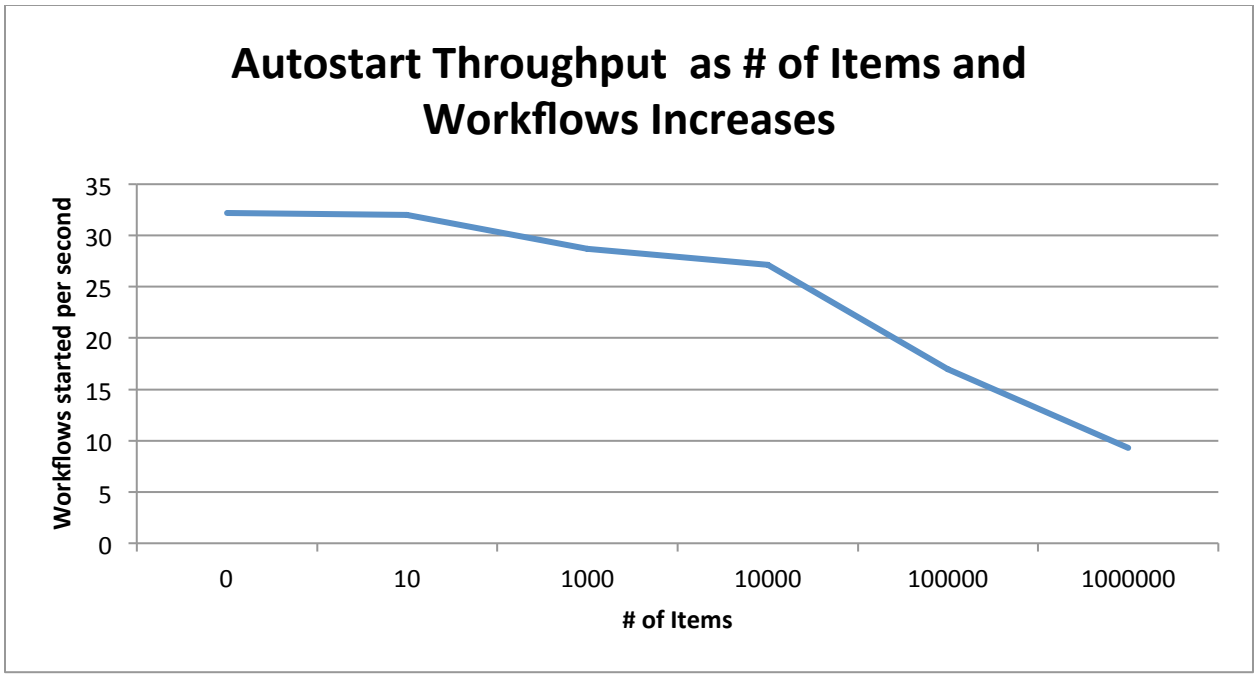



# Effect of List Size and Number of Workflow Instances on Throughput

The test in the following table shows how throughput changes as list size and number of workflows increases. Data population was done by running the auto-start workflow test continuously until 1 million items were created in the list, and stopping at different checkpoints throughout to perform throughput measurements like we did with the core throughput tests. Tests were done on a 4x1 topology.

To maintain reliability during data population, we had to keep workflow queuing on to avoid reaching the maximum number of SQL connections against the back end server. If no SQL connections are available and a workflow operation cannot to connect to the content database, the operation will not be able to run. See the "Recommendations" section for more information on workflow queuing.

| Number of Items/Workflows | Baseline solution Max (RPS) |
|---|---|
| **0** | 32.18 |
| **10** | 32 |
| **1000** | 28.67 |
| **10000** | 27.16 |
| **100000** | 16.98 |
| **1000000** | 9.27 |




For a single list and single task and history list, throughput decreases steadily between 1000 and 100000 items, but the rate of degradation reduces after that point. We attribute degradation of throughput to many factors.

One factor is the number of rows added to many tables in the content database per instance. As mentioned earlier, workflows create several list items as well as event receivers that each workflow instance registers. As table sizes grow large in different scopes, adding rows becomes slower, and the aggregate slowdown for these additions adds up to a more significant degradation than just list item creation.

Task list size contributes extra overhead specifically. In comparing throughput for workflows run on new lists vs. new task lists, task lists had more of an impact on performance. We believe this is attributed to task lists registering for more event receivers than the parent list items. The following chart describes the differences:

| Throughput with different list configurations (workflows started per second) | Million Item Task list | Empty Task List |
|---|---|---|
| **Million Item List** | 9.27 | 12 |
| **Empty Item List** | 9.3 | 13 |

If you know you will need to run large numbers of workflows against large lists and need more throughput than what your tests show you can get, we recommend thinking about whether task lists can be separated between workflow associations.

# Recommendations

This section provides general performance and capacity recommendations. Use these recommendations to determine the capacity and performance characteristics of the starting topology that you created http://technet.microsoft.com/en-us/library/cc263199(office.14).aspx to decide whether you have to scale out or scale up the starting topology.

# Hardware recommendations

For specific information about minimum and recommended system requirements, see http://technet.microsoft.com/en-us/library/cc262485(office.14).aspx.

# Scaled-out topologies

You can increase workflow throughput by scaling out Web server computers up to four, at which point further increase will be negligible. Workflow throughput can be restricted by performance-related workflow settings (see Performance-Related Workflow Settings below).

# Estimating throughput targets

Many factors can affect throughput. These factors include the number of users, the type, complexity, and frequency of user operations. More complex workflows that perform many operations against the content database or register for more events will run slower and consume more resources than others.

The out-of-box workflow used in this test creates several entries in the content database that are built into the task activities. If you anticipate use of human workflows with small numbers of tasks, you can expect similar throughput characteristics. If most workflows are very lightweight operations, throughput may be higher. But if your workflows will consist of large numbers of tasks or intense back-end operations or processing power, you can expect throughput to drop.

In addition to understanding what the workflows will do, keep in mind where the workflows will be running and whether or not they will run against large lists, on which throughput will degrade over time.

SharePoint Server 2010 can be deployed and configured in a wide variety of ways. As a result, there is no simple way to estimate how many users can be supported by a given number of servers. Therefore, make sure that you conduct testing in your own environment before you deploy SharePoint Server 2010 in a production environment.

# Workflow Queuing and Performance-Related Settings

Workflow uses a queuing system to control workflow-related stress on farm resources and the content database. With this system, when the number workflows actively executing against a database reaches an administrator-configured threshold, subsequent workflow operations are added to the queue to be run by the Workflow Timer Service, which picks up a batch of workflow work items through timer jobs every few minutes.

Several farm administrator settings directly and indirectly related to the queuing mechanism impact the performance and scale for workflow. The following sections describe what these settings do and how to adjust them to tune your performance needs.

## Understanding the Basic Queue Settings

Farm administrators can adjust the following settings to modify basic characteristics of the queuing system:

- **Workflow Postpone Threshold (Set-SPFarmConfig –WorkflowPostponeThreshold <integer>)** The maximum number of workflows that can be actively executing against a single content database

before subsequent requests and operations get put on the queue. Queued workflows show a status of "Starting". This is a farm-wide setting with a default value of 15. Note that this represents the number of workflow operations actively being processed at a time, not the maximum number of workflows that can be in progress. As workflow operations finish, subsequent operations will be able to run.

- **Workflow Event Delivery Batch Size (Set-SPWorkflow –BatchSize <integer>)** The Workflow Timer Service is an exception to the postpone threshold limit and will retrieve batches of items from the queue that can be larger than the postpone threshold and execute them one at a time. The number of work items that it picks up per run is set with the BatchSize property. The batchsize property can be set per service instance. The default value is 100. Note: when running on application servers that are not configured to be front-ends, the timer service requires workflow configuration settings in web.config to be set in the configuration database. This needs to be done through a script that calls the UpdateWorkflowConfigurationSettings() on the SPWebApplication object, which will copy the web.config settings from a front-end server.

- **Workflow Timer Job Frequency (Set-SPTimerJob job-workflow –schedule <string>)** The frequency that the workflow timer service runs can be adjusted through timer job settings. By default, it is set to run every five minutes, meaning that there can be a five minute delay before the work items at the top of the queue get processed.

  Note: Scheduled work items such as task due date expirations are also picked up by the same timer mechanism, so they will be delayed by the same time interval.

The Workflow Timer Service can be turned off on a per machine basis through the Shared Service administration experience in Central Administration. By default, it will run on every front-end computer in the farm. Each job will iterate through all the web applications and content databases in the farm.

The combination of the postpone threshold, batch size, and timer frequency can be used to limit workflow operations against the database. Your maximum throughput will be impacted by how quickly operations get queued and processed from the queue.

For example, with the default settings with a single timer service and a single content database, if there are 1000 items in the queue, it will take ten timer job runs to execute them all, which will take 50 minutes to execute. However, if you set the batch size to 1000 and set the timer job to run every minute, the operations would all begin execution after a minute. If you set the postpone threshold higher, more operations will run synchronously, reducing the number of requests that get queued and decreasing the total amount of time required to process those workflows.

Note: We recommend setting the postpone threshold no higher than 200, as concurrent workflow instances run in their own threads and will each open new SQL connections, thus potentially hitting the maximum SQL connection limits on the back end server over time.

If you do not want workflows running on front-ends and know that operations do not have to happen immediately, you can isolate the workflow timer service to run on select application servers, set the postpone threshold to a very low number to force workflows to almost always run in the timer service, and set the batch size large so that it picks up items more quickly and frequently. If you want to make sure workflows run more synchronously across the system, set the postpone threshold higher so that workflows do not get postponed often and have a more immediate impact.

Modify these settings to optimize for how you want workflows to operate. We recommend experimenting with different settings and testing them to optimize them for your environments and needs.

## Adjusting Settings for Queue Scale

If the farm will sustain heavy workflow load for long periods of time or there will be many delay events queued from workflows in the system, the number of queued workflow operations will grow. In addition to the basic queue settings above, you may need to tune the following settings to keep the queue in good health:

- **Work Item Event Delivery Batchsize**
  The table that workflow uses for queued events is a general work item table that is shared with other non-workflow features in SharePoint. Thus, there is another timer job that dequeues *non*-workflow work items. Similar to the workflow event delivery batchsize, the *work item* event delivery batchsize specifies the number of non-workflow work items that are dequeued at a time.

- **Workflow Failover Timer Job Frequency**
  In rare circumstances, if workflow events cannot be delivered to a workflow instance, the event delivery will be scheduled on the queue as a "failover" work item to be retried at a later time (starting with 5 minutes later, then 10 if it fails again, then 20, and so forth). A workflow failover timer job dequeues failover work items, and this setting adjusts the frequency that the failover timer will run. By default, this runs every 15 minutes.

- **Workflow Failover Batchsize**
  Similar to the workflow and work item batchsize settings, this setting controls the number of failover events that each failover timer job will dequeue.

Because there are many timer jobs that operate on the same table, it is possible that large numbers of queued items can cause database contention and hence reduce throughput and reliability. To reduce contention, we recommend the following:

- Balance Postpone Threshold and Workflow Batchsize such that batch size is small enough or timer job frequency high enough that a timer job can complete before the next timer job starts to avoid building up too many parallel timer job runs that cannot complete.

- To avoid table locks, do not set either of the two batch size settings higher than 5000.

**Offset the frequency of the workflow, work item, and failover timer jobs so that they are not always executing at the same times. In our data population scripts to get a large list with workflows, 4 minutes for the workflow timer job and 6 minutes for the failover worked well.**

# Improving scale for task and history lists

Workflows generate many tasks and history items per instance. These lists are indexed by default to help them scale, but as these lists grow, performance will inevitably degrade. To reduce the rate of degradation, we recommend keeping separate history and task lists for different workflow associations, and periodically changing these lists in the workflow association settings as lists become large.

Workflow also has a daily timer job (job-workflow-autoclean) that will automatically delete workflow instances and tasks for instances that have been in a completed state for more than 60 days. We recommend leaving this timer job on to keep the task lists and events on the task list as clean as possible. If data needs to be preserved, write the data to other lists or archive locations. Note: workflow history items are not deleted by this timer job. If you need to clean these up, this should be done with a script or manually through the list user interface.

## Other considerations

Removing columns on lists causes a database operation proportional to the number of items in the list. Removing workflow associations will remove the workflow status column from the list, causing a large operation on large lists. If you know that a list has millions of items, set the workflow to "No New Instance" rather than removing workflows.

# Troubleshooting performance and scalability

| Bottleneck | Cause | Resolution |
|---|---|---|
| **Database contention (locks)** | Database locks prevent multiple users from making conflicting modifications to a set of data. When a set of data is locked by a user or process, no other user or process can modify that same set of data until the first user or process finishes modifying the data and relinquishes the lock. | To help reduce the incidence of database locks, you can:<br>Distribute workflows to more document libraries.<br>Scale up the database server.<br>Tune the database server hard disk for read/write.<br>Methods exist to circumvent the database locking system in SQL Server 2005, such as the NOLOCK parameter. However, we do not recommend or support use of this method due to the possibility of data corruption. |
| **Database server disk I/O** | When the number of I/O requests to a hard disk exceeds the disk's I/O capacity, the requests will be queued. As a result, the time to complete each request increases. | Distributing data files across multiple physical drives allows for parallel I/O. The blog SharePoint Disk Allocation and Disk I/O contains much useful information about resolving disk I/O issues. |
| **Web server CPU utilization** | When a Web server is overloaded with user requests, average CPU utilization will approach 100 percent. This prevents the Web server from responding to requests quickly and can cause timeouts and error messages on client computers. | This issue can be resolved in one of two ways. You can add additional Web servers to the farm to distribute user load, or you can scale up the Web server or servers by adding higher-speed processors. |

## Web servers

The following table shows performance counters and processes to monitor for Web servers in your farm.

| Performance counter | Apply to object | Notes |
|---|---|---|
| **Processor time** | Total | Shows the percentage of elapsed time that this thread used the processor to execute instructions. |
| **Memory utilization** | Application pool | Shows the average utilization of system memory for the application pool. You must identify the correct application pool to monitor.<br>The basic guideline is to identify peak memory utilization for a given Web application, and assign that number plus 10 to the associated application pool. |

## Database servers

The following table shows performance counters and processes to monitor for database servers in your farm.

| Performance counter | Apply to object | Notes |
|---|---|---|
| **Average disk queue length** | Hard disk that contains SharedServices.mdf | Average values greater than 1.5 per spindle indicate that the write times for that hard disk are insufficient. |
| **Processor time** | SQL Server process | Average values greater than 80 percent indicate that processor capacity on the database server is insufficient. |
| **Processor time** | Total | Shows the percentage of elapsed time that this thread used the processor to execute instructions. |
| **Memory utilization** | Total | Shows the average utilization of system memory. |

# Related Resources

Workflow Scalability and Performance in Windows SharePoint Services 3.0