

mental queue™ : Functional Overview

White Paper

Document version 1.0
July 27, 2006

Copyright Information

Copyright © 1986-2006 mental images GmbH, Berlin, Germany.

All rights reserved.

This document is protected under copyright law. The contents of this document may not be translated, copied or duplicated in any form, in whole or in part, without the express written permission of mental images GmbH.

The information contained in this document is subject to change without notice. mental images GmbH and its employees shall not be responsible for incidental or consequential damages resulting from the use of this material or liable for technical or editorial omissions made herein.

mental images®, mental ray®, mental matter®, mental mill™, mental queue™, mental q™, mental world™, mental map™, mental earth™, mental mesh™, mental™, Reality™, RealityServer®, RealityPlayer®, RealityDesigner®, MetaSL™, Meta™, Meta Shading™, Meta Node™, Phenomenon™, Phenomena™, Phenomenon Creator™, Phenomenon Editor™, Phenomill™, Phenograph™, neuray™, iray®, imatter®, Cybernator™, 3D Cybernator™, Shape-By-Shading™, SPM®, NRM™, and rendering imagination visible™ are trademarks or, in some countries, registered trademarks of mental images GmbH, Berlin, Germany.

All other product names mentioned in this document may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

Table of Contents

Introduction	1
Job Management: mqmanager and mqnode	2
Job Types	2
Task Allocation	3
Application Installation	3
Manager Interface: mqcommand and mqmonitor	4
Additional Services Provided by mqmanager	4
User Management	4
Web Services API	4

Introduction

Production and computing service pipelines require the execution of software processes on a network of computing nodes to accomplish jobs such as frame rendering, special effects processing, color correction, compositing, and other generic pre- and post-processing jobs. The management of such jobs involves issues which can be difficult and time-consuming to address, such as job priority handling, failure detection and recovery, efficient distribution of tasks across the network, interdependent processes and networks comprised of computing nodes with different platforms and operating systems.

mental queue™ is a multi-purpose generic process management system which provides an effective solution to these problems. It distributes work evenly over a grid of computing nodes in an efficient and reliable manner, and provides flexible ways to configure, control and customize the system. It is able to install all necessary software on the grid of computing nodes, relieving the user of the burden of doing so. In addition, since mental queue is a scheduler, it can handle asset management, including asset allocation, online and offline transfers, and backup. Features of mental queue include:

Efficiency and Reliability

- Multi-threaded, event-driven kernel for parallel processing and efficient use of computing resources
- Process management which allows computing nodes to efficiently use CPU for allocated tasks
- Dynamic task allocation which distributes work in group sizes based on network performance
- Runtime environments to reliably propagate configurations to computing nodes
- Detection of computing node failure and reallocation of tasks to ensure job completion
- State recovery of the job queue manager in case of machine failure

Customization and Control

- Generic job type creation and support for mental ray® rendering jobs
- Support for custom process control scripts written in any language
- Choice of fixed or dynamic grouping of tasks
- Command line and GUI control from any machine on the network
- User management system to create and maintain accounts and to determine access
- Web Services API for building custom tools to monitor and audit job progress
- Support for heterogeneous networks of computing nodes
- Control of multiple hardware configurations from a single interface

mental queue consists of the following four components:

mqmanager	maintains the job queue and allocates tasks for nodes to process
mqnode	executes tasks given to it and monitors progress of execution
mqcommand	command line interface to control process management
mqmonitor	graphical user interface to control process management

The remainder of this document explains these components and their capabilities in more detail.

Job Management: mqmanager and mqnode

The mqmanager and mqnode components handle jobs and their execution. Figure 1 shows an overview.

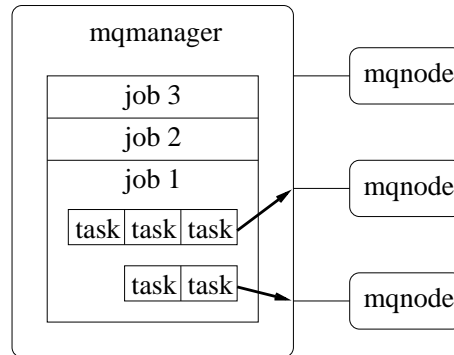


Figure 1: Overview of mental queue job management

The mqmanager component prioritizes submitted jobs, splits them into tasks, and distributes the tasks to computing nodes for processing. It monitors progress across the grid of nodes being used and adjusts as necessary to make efficient use of the grid. If a node fails, the tasks assigned to it are reallocated to ensure their completion. If the mqmanager itself fails, mental queue recovers its state upon start. The mqmanager can be installed on any computer on the grid. For grids with fewer than 10 computing nodes, the mqmanager may even be installed on a computer which is also used as a computing node without significantly impacting performance. The mqmanager component uses multi-threading techniques for efficient processing, and it handles all of the overhead required to manage the job queue.

Computing nodes are added to the grid by installing the mqnode component on them. When started, mqnode will either connect directly to a pre-defined mqmanager or broadcast a call to find an available mqmanager. Once the mqmanager is notified that a new node is available, it connects to the node and initializes its state by profiling its hardware and platform information. Then the mqnode waits for the mqmanager to assign it tasks. In order to process tasks, the mqnode needs the tasks themselves, along with any scripts and applications needed to execute it.

Job Types

Jobs are submitted to mental queue under a job type that defines the process which will be used to execute it. Two types of jobs are available in mental queue, namely generic processes and mental ray jobs.

Generic processes provide custom process control. They allow any process which is controlled by command line arguments to be used. Such processes are specified by a user-supplied script, which may be written in any language such as shell scripts, Perl, or Python. The generic process job type supports the execution of groups of tasks with a single instance of the process being used, provided that the process reports progress on the completion of each task. Tasks can also be handled one at a time, meaning that a new instance of the process is launched for each task performed. A translate option converts local paths to network paths so that files are accessible by all computing nodes, even those of different platforms and operating systems.

A mental ray® job type is also provided, which can be used to specify, edit and render scenes. The user can determine command line options, output path, and log output path, as well as activate distributed frame rendering. Just as with generic processes, local paths can be converted to network paths so that scene files are accessible by all computing nodes.

Task Allocation

Jobs in the queue are split into tasks by the mqmanager. Since some processes can use data computed for one task to process a subsequent one, grouping tasks together into blocks for one computing node to process sequentially can result in significantly better performance. The mqmanager groups tasks together to take advantage of this fact. Two types of groupings are available. Fixed allocation groups tasks into blocks which have the same user-given number of tasks in each block, while dynamic allocation groups tasks into blocks of variable size, depending on the node performance detected by the mqmanager. Figure 2 shows fixed and dynamic task allocation.

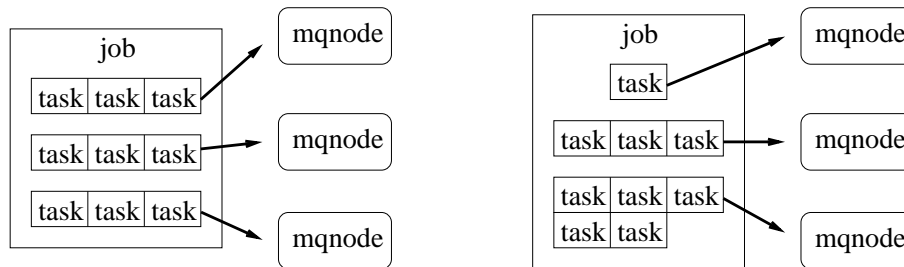


Figure 2: Fixed Task Allocation (left) and Dynamic Task Allocation (right)

Application Installation

A binary of the application, or applications, needed to execute a task, as well as any ancillary files, must be installed on the computing node which is processing it. There are two ways to ensure that the necessary binaries are installed, either by installing them in advance or by using a runtime environment. One choice for advance installation is to mirror the nodes, in other words to enforce that all nodes on the grid are identical except for network name and IP address. In this case, each node has the same operating system and identical copies of all necessary binaries and ancillaries. For heterogeneous grids of computing nodes, use of a runtime environment is recommended. A runtime environment contains all settings and binaries for all platforms and applications in use, and installs them as needed. Any settings or binaries on computing nodes are ignored, and the ones designated by the runtime environment are used instead. Figure 3 shows what is contained in a runtime environment.

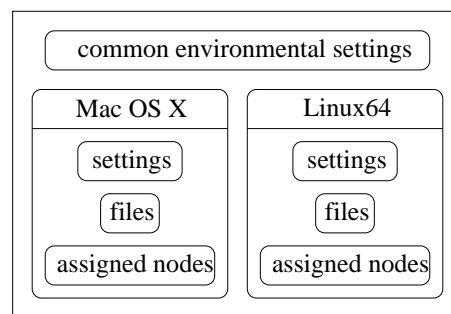


Figure 3: Runtime Environment

Manager Interface: mqcommand and mqmonitor

mental queue provides two interfaces to the mqmanager. The mqcommand component is a command line interface, and the mqmonitor component is a cross-platform, graphical user interface. The mqcommand component gives users the means to integrate mental queue into their own processes, while the mqmonitor component provides richer, high-level access to mental queue. Both can be installed and run from any machine on the mental queue grid, and any number of instances can access the mqmanager simultaneously. Computing nodes on the grid may have different operating systems, and can be controlled from a single interface which includes transparent handling of operating system nuances and path translations.

The mqcommand and mqmonitor components can be used to get information from the mqmanager and to control it, including information related to jobs, tasks and nodes to customize the entire system. Access to jobs includes the ability to perform tasks such as view job status and export the job log, submit, copy, delete, pause, stop, resume, refresh and restart jobs, edit jobs and their dependencies, and modify job priority. A mental ray job type is fully supported, and new job types can be created. When jobs are split into tasks, the tasks can be bundled together, either in groups of fixed number or dynamically, for efficiency. Node management allows the user to create, view and modify node settings and properties, and to clear, refresh, suspend, resume and restart nodes.

Additional Services Provided by mqmanager

In addition to job queue management, the mqmanager component provides other services.

User Management

mental queue user access and rights are controlled from a single administrator account. A default guest account type is provided, and the administrator can create other account types to give specific sets of permissions to users and to track user participation and success rates. A system to reward users who consistently submit jobs that complete without errors is available. When used, it gives jobs submitted by such users higher priority than jobs from users who have a history of submitting jobs with problems such as bad paths or missing assets.

Web Services API

In addition to its job handling capabilities, the mqmanager contains a Web Services API that can be used to build custom monitoring and reporting web tools. Such tools can be accessed from a web browser on any connected machine, and allow for dynamic access and control of the mqmanager. The Web Services API includes functions to access mqmanager data directly, as well as a simplified set of HTTP services to enhance tools by serving standard HTML and other documents. This allows the mental queue administrator to choose how and which aspects of job queue management are exposed to users.