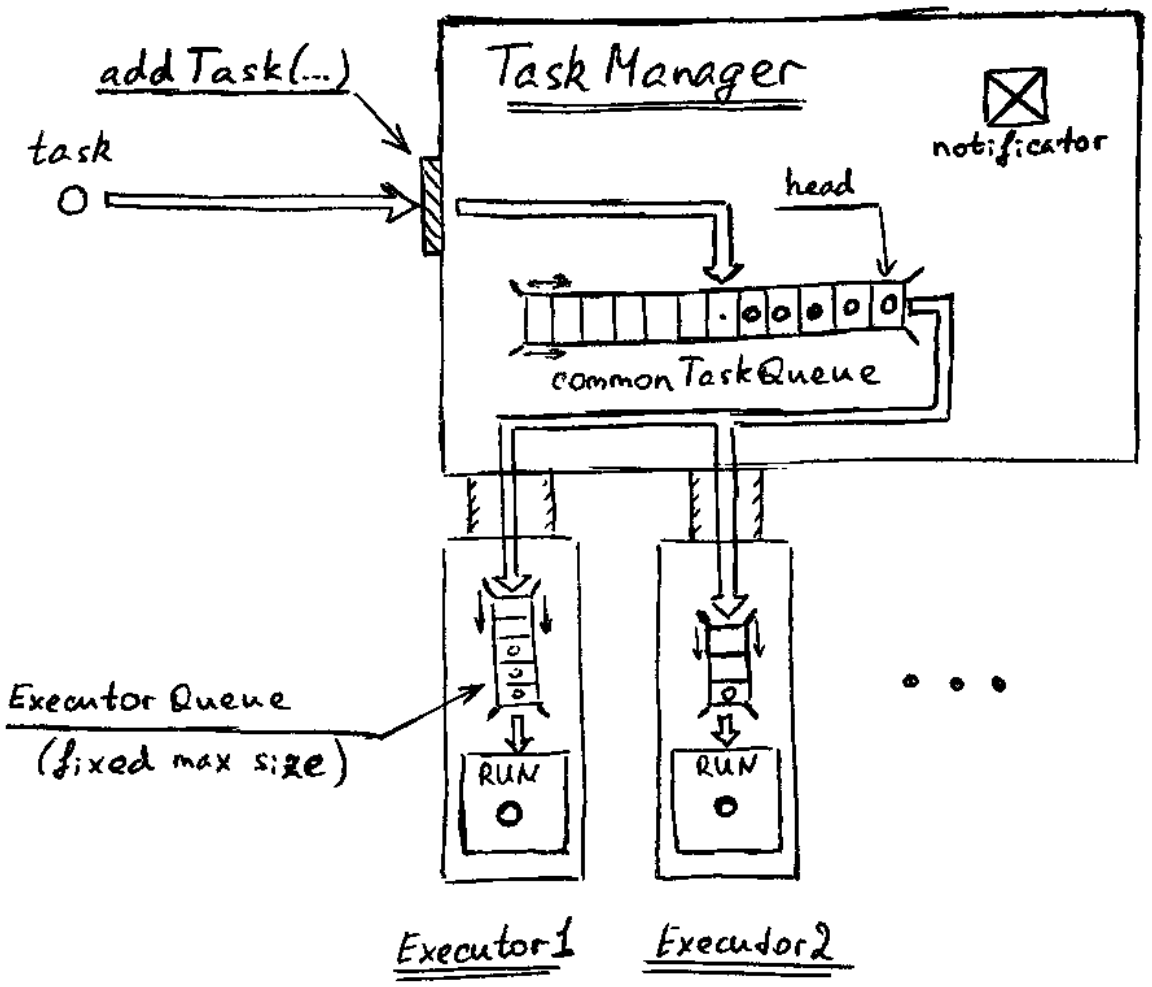
Вид разрабатываемой модели (рис.1):



Статическое описание модели:

- Имеется несколько Исполнителей (Executor1, Executor2,…).
- Каждый Исполнитель способен выполнять одновременно 1 задачу.
- Каждому Исполнителю принадлежит очередь задач (Executor Queue) фиксированной длины.
- Существует Менеджер задач (TaskManager), к которому прикреплены:

    1) Общая очередь задач (commonTaskQueue)

    2) Все исполнители

- Менеджер задач поддерживает вход (addTask), с помощью которого можно добавлять новые задачи в систему.

Динамическое описание модели:

- На вход «addTask» Менеджера задач поступает новая задача. Менеджером задач она помещается в общую очередь задач(commonTaskQueue).
- Если в общей очереди задач есть задачи, то Менеджер задач находит Исполнитель, «занятость» которого минимальна, извлекает первую задачу из Общей очереди задач и помещает ее в Очередь задач найденного Исполнителя. Если же в Общей очереди задач нет, либо все Очереди задач всех Исполнителей заняты полностью, то Менеджер ничего не делает.
- Если в Очереди задач Исполнителя есть задачи, то исполнитель извлекает первую из них и исполняет ее. После исполнения задачи Исполнитель повторяет операцию. Если же очередь пуста, то Исполнитель ничего не делает.

Описание модели на уровне потоков:

- Каждый Исполнитель – отдельный поток
- Менеджер задач – отдельный поток.
- Новые задачи, поступающие из-вне, вводятся из отдельного потока.

Преследуемые цели:

- Максимальное избавление от «холостого хода» с целью разгрузки аппаратных вычислительных средств. То есть если элемент системы вынужден простаивать (когда у Исполнителя нет задач либо когда Менеджер не может выполнить распределение), то его поток следует вводить в режим «ожидания» методом wait().
- Каждый Исполнитель может иметь произвольную максимальную длину своей Очереди задач.
- Должен быть поддержан способ остановки выполнения всех элементов системы.
- Минимизация объектов синхронизации
- Минимизация блокировок

Решения:

- Общая очередь задач и Очередь задач каждого Исполнителя являются общими ресурсами данной системы (т.е. ресурсы, к которым происходят обращения из разных потоков). Поэтому было решено при обращении к Общей очереди задач синхронизироваться по объекту Notificator (находится в Менеджере). А при обращении к Очереди задач исполнителя синхронизироваться по объекту Очереди задач Исполнителя.
- Исполнитель уходит в режим «ожидания», подписываясь на объект своей очереди задач. Менеджер задач, после добавления новой задачи в очередь задач данного Исполнителя, через метод notifyAll(), вызванный из очереди задач Исполнителя, разбудит спящего Испонителя и тот сможет продолжить работу.

- Менеджер задач уходит в режим «ожидания», подписываясь на объект Notificator. Менеджер задач должен быть разбужен, если что-то в системе изменилось: Исполнитель решил задачу либо добавлена новая задача в Общую очередь. В этом случае он начинает работать до тех пор, пока снова не будет вынужден уйти в режим «ожидания». Следовательно, Исполнитель, после того, как изъял задачу из своей очереди, должен «разбудить» Менеджера. Аналогично должен поступить метод addTask(…) после добавления новой задачи в Общую очередь.
- За меру «занятости» Исполнителя в реализации программы принята относительная занятость Очереди задач в данный момент времени.

Алгоритм:

Поток Менеджера задач:

1. Найти Исполнителя с минимальной загруженностью. Если такого нет (все заняты), то [уснуть] на Notificator-е. Если есть, то [2].
2. Если в Общей очереди задач нет, то [уснуть] на Notificator-е. Если есть, то (с синхронизацией по Notificator) забрать из общей Очереди задач первую задачу, затем (с синхронизацией по Очереди задач Исполнителя) поместить эту задачу в Очередь задач Исполнителя и послать уведомление Исполнителю на случай, если тот «спит». Перейти к [1]

После всех ожиданий и в начале общего цикла проверять, нужно ли продолжать работу (проверка флага). Если не нужно, то прекращать исполнение.

Поток Исполнителя:

1. Проверить, если ли задачи в Очереди задач исполнителя. Если нет, то [уснуть] на своей очереди задач. Если есть, то (с синхронизацией по Очереди задач Исполнителя) забрать первую задачу.
2. Затем (с синхронизацией по Notificator) уведомить Менеджера об изменении своего состояния на случай, если тот «спит».
3. Исполнить задачу. Перейти к [1].

Внешний поток, вносящий новую задачу:

1. (с синхронизацией по Notificator) добавить новую задачу в Общую очередь задач и уведомить Менеджера об изменении состояния системы на случай, если тот «спит».

Проблемы реализации и их решения:

- При поиске минимально загруженного Исполнителя, Менеджер выполняет только операцию чтения из общих ресурсов (из Очередей задач Исполнителей). Чтобы не добавлять лишних блокировок, было решено делать это без синхронизации. Проблема такого подхода в том, что к моменту обхода состояние очередей может измениться (а именно уменьшиться). То есть может быть неверно определен Исполнитель с минимальной загруженностью. Но при интенсивной работе системы такая ситуация совершенно не критична, поэтому допустима. Однако может произойти следующее: во время обхода Менеджер увидел, все очереди заняты. К концу обхода одна из очередей освободилась, но Менеджер уже совершил обход и решил уснуть. Это может заблокировать систему. В этом случае решено было использовать дополнительный флаг «неизменности очередей исполнителей». Перед обходим этот флаг устанавливается Менеджером в FALSE. Во время обхода Менеджера какой-либо Исполнитель, при изъятии новой задачи, устанавливает данный флаг в TRUE. После обхода Менеджер, уверенный в том, что все занято, (с синхронизацией по Notificator) проверяет данный флаг. Если тот TRUE, значит «спать» нельзя, и следует заново все проверить. Если FALSE, то можно законно «спать».
- Чтобы избежать взаимоблокировок, в блоках синхронизации не вызываются методы, способные заблокировать поток без отпускания объекта синхронизации. (метод wait() блокирует, но отпускает объект синхронизации). И в блоках синхронизации не вызываются вложенные блоки синхронизации.
- Число объектов синхронизации равно 1+[кол-во Исполнителей]. Уменьшение этого числа может привести к взаимоблокировкам Исполнителей, чего было решено избежать.

Идеи и предложения:

- Возможно создание дополнительного класса Строителя, основной функцией которого было бы «создание», «сборка» и «разборка» вышеописанной системы по требуемым характеристикам.
- По-хорошему роль Notificator-а может исполнять и Общая очередь задач, так что объект Notificator избыточен, хотя и удобен в использовании.
- Оценку «занятости» Исполнителя можно осуществлять и любыми другими способами, например:
    - Кол-во стоящих в очереди задач
    - Зависимость от приоритета потока Исполнителя
    - Зависимость от каких-либо дополнительных характеристик Исполнителя
    - и т.п.
- Концепция «усыпления» потоков не обязательна. В теории можно без остановки выполнять прогон по рабочему циклу этих потоков. При наличии большого кол-ва наипростейших задач это могло бы и стать эффективней, но если задач будет недостаточно много или задачи будут хоть немного сложней, холостые прогоны по рабочему циклу начнут приносить серьезные убытки по производительности.