Отчёт

ВШЭ

2 апреля 2018 г.

Содержание

1 Реализация дискретно-событийной библиотеки Simlibrary для моделирования СХД

1.1 Описание Environment имитации

Имитация системы происходит при помощи объекта типа Environment, который полностью отражает текущее положение системы. Данный объект обладает следующими полями:

currentTime Текущее время системы типа $float64$. Изменяется дискретными шагами.

workers Словарь типа $map[uint64] * Process$, где в качестве ключа выступает идентификатор PID объекта, а в качестве значение указатель $Worker$.

routesMap Словарь типа $map[Route] * Link$. Содержит значения обо всех путях, возможной в данной конфигурации компьютерной сети. $Route$ – структура, имеющая в качестве своих полей $start$ и $finish$ – указатели на хост, которые являются началом и концом пути, соответственно. Значение $*Link$ – является указателем на сеть, по которой будет проходить передача пакетов в обе стороны.

queue Поле типа $eventQueue$. Является глобавльным хранилищем всех событий во время имитации сложных сложных систем.

mutex Поле типа $sync.Mutex$. Примитив синхронизации нужный для того, чтобы обеспечивать консистентность доступа к данным, таким как очередь событий $queue$.

shouldStop Поле типа $bool$. Во время прогона симуляции является равным 1. После того, как все события в имитации заканчиваются, либо при наличии специального события, выставляется в отрицательное значение и прогон прекращается.

hostsMap Словарь типа $map[string]HostInterface$. Содержит информацию обо всех хостах, которые имеются в симуляции. В качестве ключа словаря – имя хоста, в качестве значения интерфейс типа HostInterface, который обобщает такие типы как $host$, $NetworkSwitch$ и $IOBalancer$.

vesninServers Поле типа $[]*Host$. Является списком, который содержит информацию о рабочих дисковых контроллерах, поддреживающих отношения с клиентом, представленных в виде указателя на $*Host$.

allVesninServers Поле типа $[] * Host$. Поле типа $[] * Host$. Является списком, который содержит информацию о обо всех (рабочих и нерабочих) дисковых контроллерах, поддреживающих отношения с клиентом, представленных в виде указателя на $*Host$.

storagesMap Словарь типа $map[string] * Storage$. В данном контейнере хранится информация обо всех дисках, примонтированных к системе хранения данных. В качестве ключа словаря используется идентификатор конечного хранилища данных, а качестве значения – указатель на дисковое хранилище.

linksMap Словарь типа $map[string] * Link$. Контейнер, содержащий информацию обо всех сетях, представленных в данной симуляции. В качестве ключа словаря используется идентификатор сети, а качестве значения – сеть, по которой будет идти передача данных.

FunctionsMap Словарь типа $map[string]func(*Process, []string)$. Данный контейнер хранит информацию обо всех фукнциях, которые будует

запущены в качестве горутин в начальный момент времени (время запуска симуляции). Функции должны быть объявлены в файле deployment.xml. В качестве ключа словаря используется идентификатор функции, представленный в строковом виде, а качестве значения – указатель на функцию.

daemonList Поле типа $[] * Process$. Является списком, который содержит информацию о горутинах, которые во время симуляции СХД, являются представлениями Unix-демонов и самостоятельно должны завершить своё исполнение.

pid Поле типа $ProcessID$. Указатель на функцию, которая исполняется в текущий момент времени.

waitWorkerAmount Поле типа $uint64$. Количество горутин, запущенных в текущий момент времени, завершения которых нужно ожидать для того, чтобы наполнить очередь актуальными текущими событиями.

stepEnd Поле типа $chaninterface$. Является средством коммукации горутин с главной ($master$) горутиной. В данный канал связи горутины, которые исполняются в текущий момент времени, сигнализируют о своём завершении.

nextWorkers Поле типа $[] * Process$. Является списком, который содержит информацию о горутинах, которые должны буть запущены на следующем шаге работы имитации системы со статусом OK.

timeOutWorkers Поле типа $[] * Process$. Является списком, который содержит информацию о горутинах, которые должны буть запущены на следующем шаге работы имитации системы со статусом $TIMEOUT$.

anomalyWorkers Поле типа $[] * Process$. Является списком, который содержит информацию о горутинах, которые должны буть запущены на следующем шаге работы имитации системы со статусом $FAIL$.

logsMap Словарь типа $map[string]float64$. Данный контейнер хранит информацию, которая впоследствии будет выведена в виде логов системы. В качестве ключа словаря используется идентификатор наблюдаемого значения, а качестве значения – числовая характеристика данной величины.

unitsMap Словарь типа $map[string]float64$. Данный контейнер хранит информацию о единицах системы измерений, принятых в данной симуляции. В качестве ключа словаря используется идентификатор единицы измерения, а качестве значения – численная характерстика относительно эталона.

backupRoutesMap Словарь типа $map[Route] * Link$. Содержит значения обо всех запасных (backup) путях, возможной в данной конфигурации компьютерной сети. $Route$ – структура, имеющая в качестве своих полей $start$ и $finish$ – указатели на хост, которые являются началом и концом пути, соответственно. Значение $*Link$ – является указателем на сеть, по которой будет проходить передача пакетов в обе стороны.

HostLinksMap Словарь типа $map[HostInterface][] * Link$. Данный контейнер хранит информацию о сетях, к которым имеет доступ каждый хост. В качестве ключа словаря используется идентификатор хоста, а качестве значения – список, состоящий из указателей на сеть, принадлежащих данному хосту.

LinkBackupsMap Словарь типа $map[*Link] * Link$. В качестве ключа словаря используется идентификатор конечного хранилища данных, а качестве значения – указатель на дисковое хранилище.

Таблица 1: My caption

TB	1000^4 byte
GB	1000^3
MB	1000^2
KB	1000
B	1
GBps	1000^3 byte per sec
MBps	1000^2 byte per sec
KBps	1000 byte per sec
Bps	1 byte per sec
Gf	1000^3 flops
Mf	1000^2 flops
Kf	1000 flops
f	1 flops

1.1.1 Функции Environment

func NewEnvironment() *Environment

Входные аргументы: отсутсвуют.

Выходное значение: переменнная типа *Environment.

Описание функции:Создаёт и инициализует необходимые поля для функцирования симуляции, такие как:

- queue
- workers
- SendEventsNameMap
- ReceiveEventsNameMap
- ReceiverSendersMap
- stepEnd
- logsMap
- HostLinksMap
- LinkBackupsMap

func createUnits()

Входные аргументы: отсутствуют.

Выходное значение: отсутствует.

Описание функции: Инциализирует единицы измерения необходимые при симуляции системы.

func (env *Environment) stopSimulation($_E ventInterface$**)**

Входные аргументы: указатель на объект типа *Environment.

Выходное значение: отсутствует.

Описание: Даннная функция останавливает исполнение программы путем выставления флага shouldStop в положительное значение.

func (env *Environment) updateQueue(deltaTime float64)

Входные аргументы: указатель на объект типа *Environment.

Выходное значение: отсутствует.

Описание: Даннная функция обновляет очередь событий за время $deltaTime$.

func (env *Environment) CreateTransferEvents()

Входные аргументы: указатель на объект типа *Environment.

Выходное значение: отсутствует.

Описание: Даннная функция создаёт события, которые имитируют передачу данных от одного хоста к другому.

func (env *Environment) Step() EventInterface

Входные аргументы: указатель на объект типа *Environment.

Выходное значение: текущее событие симуляции.

Описание: Даннная функция осуществляет шаг симуляции, который состоит из следующих шагов.

1. **Создать события, которые имитируют передачу данных от одного хоста к другому. Проверить является ли этот шаг симуляции последним. Проверить симуляцию на возникновение дедлоков. Получить событие из очереди с минимальным значением времени. Обновить текущее время. Обновить очередь событий за время, прошедшее с времени прошлого события. Обработать коллбэки (callbaks) текущего события. Проверить является ли этот шаг симуляции последним.**

func (env *Environment) FindNextWorkers(event EventInterface)

Входные аргументы: указатель на объект типа *Environment, текущее событие EventInterface.

Выходное значение: отсутствует.

Описание: Даннная функция занимается поиском горутин, которые должны начать исполнение после выполнения текущего шага. Данный список включает в себе также горутины, которые начнут исполнение со статусами OK, $FAIL$, $TIMEOUT$.

func (env *Environment) SendStartToSignalWorkers()

Входные аргументы: указатель на объект типа *Environment.

Выходное значение: отсутствует.

Описание: Даннная функция рассылает сигналы через каналы коммуикации горутинам, которые должны начать исполнение после выполнения текущего шага. Данный список включает в себе также горутины, которые начнут исполнение со статусами OK, $FAIL$, $TIMEOUT$.

func (env *Environment) WaitWorkers()

Входные аргументы: указатель на объект типа *Environment.

Выходное значение: отсутствует.

Описание: Даннная функция дожидается выполнения задач текущими горутинами, которым были посланы сигналы на предыдущем этапе.

1.2 Описание примитивов, использованных при имитации системы

Входные аргументы: отсутствуют.
Выходное значение:
Описание функции:

1.3 Устройство очереди

Наиболее важным элементом при реализации моделирования сложных систем является поддержание консистентности очереди событий. В текущей версии библиотеки она реализована при помощи встроенного в язык программирования интерфейса "container/heap". Данный интерфейс представляет структуру данных под названием дерево, которое обладает свойством, что каждая его узел является минимальным значением в его поддереве. Эта структура данных была выбрана для моделирования, т.к является наиболее распространенной при реализации очереди событий с приоритетом, которым в случае моделирования событийных имитаций является время окончания события.

Для имплементации данного интерфейса были реализованы следующие функции:

Таблица 3: My caption

Функция	Входные аргументы	Выходные аргументы	Описание функции
func (eq eventQueue) Len() int	-	Длина очереди, тип int	Данная функция возвращает значение длины очереди.
func (eq eventQueue) Less(i, j int) bool	Индексы элементов очереди. Тип int	Тип bool	Данная функция задаёт правило сравнения и сравнивает элемент очереди с индексом i с элементом с индексом j. В случае, если первый элемент больше, то возвращается логическое да, в противном случает – логическое нет.
func (eq eventQueue) Swap(i, j int)	Индексы элементов очереди. Тип int	-	Данная функция меняет местами элемент очереди с индексом i с элементов очереди с идексом j.
func (eq *eventQueue) Push(e interface{})	Значение, которое необходимо добавить в очередь. Тип interface{}	-	Данная функция добавляет новый элемент e в очередь событий.
func (eq *eventQueue) Pop() interface{}	-	Минимальнй элемент в очереди. Тип interface{}	Данная функция извлекает минимальный элемент из очереди событий.
func(eq *eventQueue) Fix(h Interface, i int)	h - элемент в очереди, который нуждается в изменениях. i - новый приоритет	-	Данная функция меняет приоритет у элемента h в очереди событий на приоритет i.