Взаимодействие системы мониторинга со сторонним программным обеспечением*

А. Г. Тарасов
Вычислительный центр ДВО РАН, Хабаровск, Россия
e-mail: taleks@as.khb.ru

Предлагается подход к взаимодействию системы мониторинга со сторонним программным обеспечением посредством уведомлений о событиях. Приведены практические примеры осуществления такого взаимодействия и показатели производительности, позволяющие оценить масштабируемость данного подхода.

Ключевые слова: вычислительные кластеры, системы мониторинга, системы виртуализации, искусственные нейронные сети.

Введение

При разработке вычислительного комплекса (ВК) необходимо решить ряд задач. Одна из них — развертывание средств управления, а также систем мониторинга (СМ). Исследованию вопросов, связанных с организацией сбора данных мониторинга и контроля ВК, статистики использования локальных вычислительных сетей и программных комплексов, посвящен ряд работ ([1 – 3] и др.). Значительные результаты в данном направлении были достигнуты и в смежных областях: в теории управления и управляющих систем, теории автоматов, в системном анализе и системах поддержки принятия решений [4 – 7].

Задачи мониторинга на сегодня остаются сложными в практической реализации. В то же время контроль компонент программно-аппаратного комплекса и оказание управляющих воздействий являются важными критериями, необходимыми для организации высокопроизводительных распределенных вычислений. Пользователи ВК нуждаются в информации о том, как выполняется отправленное на ВК задание (приложение и описание требуемых для выполнения ресурсов) и как последнее влияет на ВК. С ростом числа вычислительных узлов в составе ВК увеличивается и объем данных, которые необходимо анализировать администратору ВК, в силу чего повышается вероятность ошибки, связанной с человеческим фактором.

Многие СМ успешно и эффективно решают ряд задач мониторинга. Несмотря на все многообразие архитектур СМ, до сих пор отсутствует возможность интеграции различных систем между собой, затруднена разработка новых функций к уже действующим СМ. В связи с этим существует потребность в создании новых подходов к архитектуре систем мониторинга, позволяющих устранить указанные недостатки.

В статье [8] были описаны расширяемая архитектура СМ и созданное на ее основе приложение Grate, представляющее собой набор служб. Разработанная архитектура

*Работа выполнена в рамках ФЦП “Научные и научно-педагогические кадры инновационной России” (проект № 02.740.11.0626) и поддержана грантом ДВО РАН № 09-I-П1-01.

дает возможность наращивать функции мониторинга, в том числе добавлять сторонние модули к уже используемым СМ, не останавливая их работу, осуществлять совместный мониторинг и анализ данных, поступающих от различных СМ. В настоящей работе основное внимание уделено вопросам взаимодействия СМ с независимо исполняемым программным обеспечением в целях повышения эффективности работы ВК.

1. Трёхуровневая архитектура системы мониторинга

Предложенная в [9, 10] архитектура может решать следующие задачи:

1 — получение от вычислительного комплекса контролируемых характеристик и сохранение их значений;

2 — выявление значимых событий на основе полученных данных;

3 — запуск средств реагирования (отклика) на значимое событие.

Под *значимым событием* будем понимать переход ВК из текущего состояния в такое, при котором происходит существенное изменение важных для пользователя СМ контролируемых характеристик. К подобным событиям относятся события, возникающие при недостатке оперативной памяти на вычислительном узле, при аномально высокой активности подсистемы ввода-вывода и др.

В указанной архитектуре системы мониторинга передача данных осуществляется лишь между соседними логическими уровнями СМ. Это позволяет изменять и дополнять уровни, не нарушая общей работоспособности системы. Одно из важных следствий такого подхода — упрощение взаимодействия СМ со сторонним программным обеспечением, что в свою очередь позволяет расширять функциональные возможности мониторинга.

Нижний уровень ответствен за сбор текущих значений контролируемых величин. Источником данных может быть какой-либо сенсор физического или логического устройства, например, SNMP-счетчик сетевого устройства или данные служб сторонних СМ. *Промежуточный уровень* отвечает за получение информации с нескольких источников данных, преобразование её к унифицированному формату, предварительный анализ значений контролируемых характеристик. Этими задачами занимаются *центры сбора данных* (ЦСД). На данном уровне функционируют и *ретрансляторы событий*. *Уровень приложений* обеспечивает полнофункциональный анализ данных и представление их пользователю. Выполняющееся на нем программное обеспечение может работать за пределами ВК, получая всю необходимую информацию от промежуточного уровня.

В процессе мониторинга вычислительного комплекса динамически формируется *иерархическая структура данных* (ИСД), узлами которой может быть любой определяемый источником данных программный или аппаратный компонент. Базовой единицей хранения данных является метрика. *Метрика* представляет собой набор значений определенного типа, отражающий текущее состояние или изменение характеристики в узле ИСД, к которому принадлежит эта метрика. Так, при мониторинге распределенной вычислительной сети (РВС) могут быть сформированы метрики РВС (например число кластеров), метрики кластеров (количество вычислительных узлов) и др.

Каждый источник данных производит измерение множества метрик, возможно неодинакового для различных источников. Через определенные промежутки времени производится мгновенный снимок состояния метрик, который передается в ЦСД, ответственный за обработку информации.

Важными элементами описываемой архитектуры СМ являются триггер и действие. *Триггер* — это логический элемент, содержащий условие или выражение, истинность которого необходимо проверить для последнего значения метрики. Если условие выполнено, то состояние триггера изменяется, что приводит к выполнению заранее определенных действий. Основная задача триггеров состоит в проверке выполнения простых условий, наложенных на значения метрик. Предполагается, что проверка триггеров в основном проводится в ЦСД.

Под *действием* понимается реакция на соответствие значений метрики условиям триггера, заключающаяся в исполнении набора инструкций. Одному триггеру может быть сопоставлено несколько действий и наоборот — одному действию — несколько триггеров.

Каждый контролируемый узел СМ может генерировать уведомление о возникшем событии (рис. 1). В процессе работы СМ приложения пользователя “подписываются” (регистрируются) на ретрансляторе событий, в качестве которого обычно используется ЦСД. При подписке указывается, в уведомлениях о каких событиях заинтересовано приложение. Центр сбора данных получает информацию о событии и при необходимости передаёт её подписчикам.

Приложение пользователя может обрабатывать, игнорировать события или генерировать новые, которые будут обработаны другими подписчиками. Если подписок на одно и то же событие несколько, то приложения получают уведомление в порядке приоритета, указанного при подписке.

Анализ метрик выполняется на третьем уровне архитектуры, на котором информация о состоянии контролируемых величин принимается от ЦСД в структурированном виде. Запрос на получение данных может содержать указания по агрегированию значений метрик: осреднять данные по всем узлам, показывать суммарную информацию и др. Распределение элементов СМ по трем слабо связанным уровням позволяет реализовать опрос различных источников данных единообразно, обеспечивая совместную работу нескольких систем. Механизмы подписки и уведомлений о событиях являются существенным отличием предложенной архитектуры от ряда аналогов, именно они обеспечивают взаимодействие СМ со сторонним программного обеспечения.

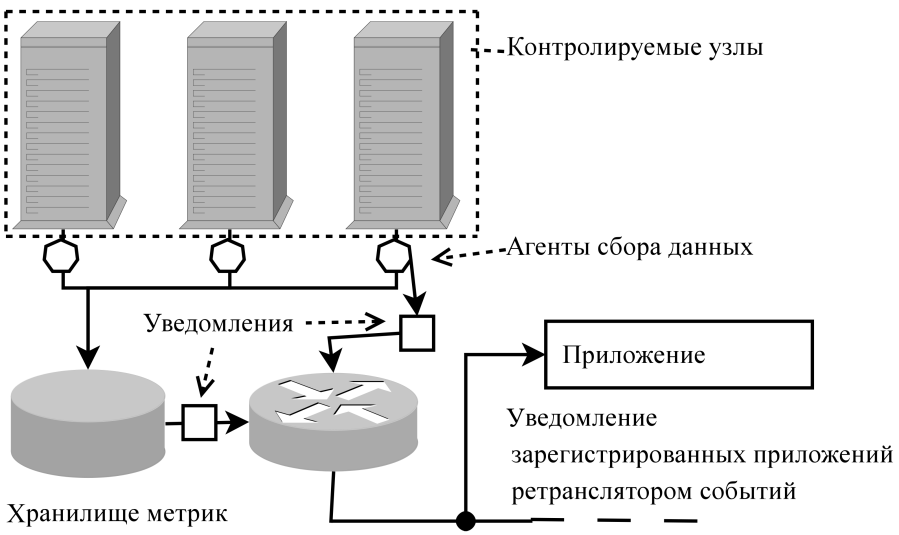

Рис. 1. Схема работы системы уведомлений

В численных экспериментах, приведенных ниже, применяется разработанная с использованием описанной архитектуры системы мониторинга Grate, состоящая из нескольких независимо функционирующих модулей, написанных на языке программирования Java. Эти модули используют общую библиотеку интерфейсов и классов, позволяющую работать с триггерами, уведомлениями о событиях и другим элементами СМ.

2. Взаимодействие со сторонним программным обеспечением

На кластере КВЦ-1, собранном в 2004 г. из неспециализированных компонент, для контроля характеристик ВК использовалась СМ Ganglia [11]. Отсутствие возможности реализовать отклик системы штатными средствами данной системы привело к необходимости добавления функции контроля негативных изменений, связанных с повышением температуры вычислительных узлов, например, в случае отказа системы активного охлаждения. Интеграция в разработанную СМ Grate выполнялась следующим образом: в модуле grated была реализована бóльшая часть функциональных возможностей службы gmetad СМ Ganglia, данные мониторинга поступали от служб gmond с предварительно настроенной метрикой температуры. Эта метрика не является стандартной для СМ Ganglia, но её можно формировать с применением стандартных служб операционных систем (ОС) и передавать через приложение gmetric, входящее в состав СМ Ganglia.

При этом проводилась проверка триггеров, были доступны и остальные функции разработанной СМ. При превышении заданной величины происходили запись в журнал событий и отправка уведомления администратору кластера по электронной почте. В критическом случае принудительное отключение узла инициировалось аппаратными средствами, однако сигнал о необходимости отключения был использован для подготовки программного обеспечения к остановке работы и отключению питания. Модуль grated проводил проверку триггеров (по одному на каждый вычислительный узел), расширяя тем самым функциональные возможности используемой СМ Ganglia. Визуализация выполнялась программным обеспечением grate-client на основе данных, поставляемых от grated, тем не менее веб-фронтэнд СМ Ganglia в штатном режиме мог обрабатывать и предоставлять доступ к данным, получаемым от grated.

Важным результатом разработки является интеграция ранее установленной системы в описываемую СМ без остановки и перенастройки источников данных. Замена агента сбора данных позволила получить сравнимую или лучшую производительность при более широких функциональных возможностях, что с использованием других доступных на момент тестирования СМ было невозможно.

Тестирование, выполненное при мониторинге до 15 узлов ВК, показало, что производительность СМ Grate (модуля grated) превышает производительность СМ Ganglia (модуля gmetad). Тем не менее в силу применения виртуальной машины Java [10] grated использует значительные объемы оперативной памяти на управляющем сервере. В настоящее время исследовать зависимость роста накладных расходов на организацию мониторинга от числа вычислительных узлов более 15 на реальном кластере не представляется возможным из-за отсутствия у автора доступа к подобным ВК для получения данных мониторинга. Вместе с тем можно оценить затраты вычислительных ресурсов при мониторинге большего числа вычислительных узлов исходя из имеющихся результатов моделирования контроля 100 вычислительных узлов по методике оценки масштабирования, примененной разработчиками СМ Ganglia.

При использовании источников данных, аналогичных СМ Ganglia (gmond), и групповой передачи данных (multicast, протокол UDP) с увеличением числа узлов рост накладных расходов на организацию сетевых подключений для получения значений метрик незначителен. Большее влияние на повышение вычислительных затрат оказывает проверка условий триггеров, количество которых зависит от числа вычислительных узлов. При проведении численных экспериментов последнее подтверждается: для восьми узлов выигрыш в производительности grated в сравнении с gmetad составляет 45 %, с увеличением числа узлов разница уменьшается по закону, близкому к линейному, приближаясь при моделировании мониторинга 100 вычислительных узлов к 10 %. Максимальные затраты на мониторинг составляют менее 0.5 % от всего времени работы центрального процессорного устройства (ЦПУ) сервера, на котором запускался grated (по данным за 41 день использования вычислительного времени процессом java, который исполнял grated).

Другая важная функция предлагаемой системы — выявление значимых событий — позволяет более полно использовать ВК за счёт сокращения не полезных затрат вычислительных ресурсов. Поскольку с ростом числа узлов и контролируемых характеристик системному администратору все сложнее в непрерывном режиме следить за изменениями значений опрашиваемых характеристик ВК, то возникает необходимость автоматизации рутинных операций. При этом многие изменения в работе ВК могут быть обнаружены и в ряде случаев предотвращены с использованием различных систем автоматизированного контроля.

Примером таких обнаруживаемых изменений могут быть последствия неэффективно работающего приложения, не использующего вычислительные ресурсы ВК полностью во время своего выполнения. Часто встречаемым примером является неверный выбор семейства функций при пересылке MPI-сообщений в программе пользователя, а также простой на операциях ввода-вывода дисковой подсистемы, что выявляется по относительно большой доле времени работы центрального процессорного устройства в пространстве ядра (полезные вычисления программ пользователя производятся ЦПУ в пространстве пользователя). Такая ситуация хорошо распознается обученной искусственной нейронной сетью (ИНС) и может быть предотвращена при своевременном вмешательстве администратора ВК или разработчика программы.

В работе кластера ВЦ ДВО РАН была зафиксирована более серьезная проблема, связанная с запросом одним из вычислительных заданий виртуальной памяти в объеме, приведшем к исчерпанию ресурса файла подкачки (рис. 2). Вследствие этого произошёл отказ ряда системных приложений, нуждавшихся в свободной оперативной памяти, следствием чего, в частности, стало отсутствие данных на графике в области A. Сопутствующим результатом было общее замедление работы всех приложений, разделяющих узел ВК со сбойным вычислительным заданием. Очевидно, такое поведение приложения недопустимо, и критическая ситуация должна быть предотвращена как можно раньше.

В работах [12, 13] в целях выявления состояния, в котором находятся вычислительный узел и ВК в целом рассмотрена возможность применения ИНС нескольких типов. Показано, что обученная ИНС обратного распространения способна верно определять текущее состояние ВК при заранее неизвестных входных данных. Это обусловило выбор модуля ИНС в качестве компоненты разработанной СМ, ответственной за выявление изменений в работе ВК.

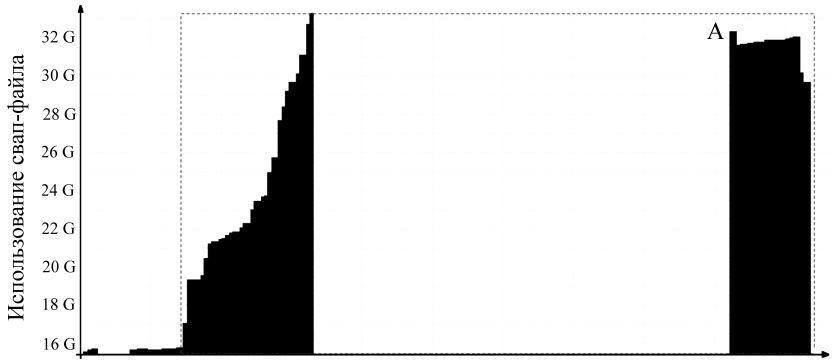

Рис. 2. Критическая ситуация на вычислительном узле

В численном эксперименте применялся кластер, состоящий из четырех узлов, под управлением операционной системы CentOS Linux v.4.4. Модуль ИНС выполнялся как независимое приложение, принимающее значения метрик от службы grated. После обработки входных данных система мониторинга получала сведения о событиях (изменениях в состоянии ВК) через механизм уведомлений. В приложение-обработчик в свою очередь поступала информация о событиях, после чего выполнялась запись в журнал и отправлялась электронная почта администратору ВК. Приложение-обработчик и служба grated CM Grate выполнялись на управляющем узле, модуль ИНС выполнялся на отдельной вычислительной станции.

В процессе предварительного исследования были выделены следующие значимые входные параметры для работы ИНС: доля свободного места в файле подкачки, отношение числа пользовательских процессов к числу ядер, доля свободной оперативной памяти, отношение загруженности процессора пользовательскими процессами к его загруженности системными процессами, доля свободной памяти на жестком диске, число принятых байт на сетевых интерфейсах, число отданных байт, а также производные от них величины. Такие параметры измерялись как для кластера в целом, так и для каждого узла в отдельности. В результате обучения методом обратного распространения ошибки была получена двухслойная ИНС, способная на основе входных данных для одного вычислительного узла выдать на выходах значения, указывающие на вероятность нахождения узла в одном из состояний. Под состоянием понимаются значения различных значимых метрик (температура процессора, количество свободной оперативной памяти, число выполняемых процессов и др.).

С целью накопления достаточно информативной обучающей выборки данных штатной работы ВК был выполнен тест HPL[1]. Модуль ИНС опрашивал состояние ВК через систему мониторинга. При этом сеть методом обратного распространения ошибки обучалась для выявления следующих основных классов состояний: неэффективное использование ЦПУ (в смысле малой доли полезных вычислений в пространстве пользователя; это состояние было смоделировано специально разработанной утилитой, поскольку получить его из данных мониторинга теста HPL невозможно), штатная работа (доля полезных вычислений до 75 %, обмен данными между вычислительными узлами невелик), стадия интенсивных вычислений (доля полезных вычислений более 75 %, значительный обмен данными между узлами ВК), наконец, простаивание вычислительного узла (загрузка ЦПУ на контролируемых узлах менее 5 %).

[1] http://www.netlib.org/benchmark/hpl

В рамках исследования взаимодействия нейросетевого модуля с СМ было выполнено более ста тестов для получения обучающей и проверяющей выборки по методике, приведенной в [12], с единственным отличием: данные мониторинга предоставлялись не выборкой из базы данных RRD (Round-robin database) СМ Ganglia, а из разработанной СМ в реальном времени. Обучение ИНС проведено при различных комбинациях входных векторов. На период тестирования (17 дней) модуль ИНС был запущен в режиме контроля экспериментального ВК, проверено более 4000 входных векторов для каждого из вычислительных узлов. В целях подтверждения повторяемости результатов проведено несколько дополнительных численных экспериментов, повторяющих методику обучения. При этом обучающая выборка, общее время эксперимента и объём участвующих в проверке данных были меньше, но результаты в численном выражении оказались близки приведенным в табл. 1, где представлены данные экспериментов с ИНС обратного распространения с различным числом нейронов.

Искусственная нейронная сеть обратного распространения при 12 нейронах и 30 циклах обучения верно отнесла текущее состояние к одному из классов, приведённых выше. При изменении класса состояния модуль ИНС посылал уведомление о событии в СМ, которая в свою очередь передавала его приложению-обработчику. Была смоделирована также и ситуация активного использования файла подкачки с помощью тестовой задачи, активно выделяющей оперативную память без возврата её системе. ИНС успешно распознала критическую ситуацию и уведомила администратора кластера.

Производительность исследуемого модуля ИНС была достаточна для обработки 500–1000 входных векторов в сек, включая накладные расходы на получение данных мониторинга (порядка 1 Кб на один вычислительный узел), формирование уведомлений о событии смены состояния и прочие вычислительные затраты, не связанные с расчётами ИНС. Такая производительность программного обеспечения позволяет осуществлять в реальном времени мониторинг ~ 500 узлов ВК, если все значения метрик одного узла приходят не чаще, чем один раз в сек (что согласуется с практикой: обычно СМ настроены получать значения метрик не чаще, чем один раз в 30–60 сек).

Для определения изменений в работе ВК в автоматизированном режиме использовались ИНС с относительно несложной топологией. Модуль ИНС можно модифицировать, не переписывая при этом исходного кода СМ Grate. Более того, уведомления о событиях и данных мониторинга могут обрабатываться одновременно несколькими приложениями (или, например, независимыми модулями ИНС), что позволяет более полно анализировать состояние ВК с применением в том числе и других методов и алгоритмов.

Другой задачей, связанной с повышением эффективности работы ВК, является перенос вычислительной нагрузки с одних вычислительных узлов на другие. Данная опе-

Т а б л и ц а 1. Доля верно распознанных векторов

Число нейронов	Количество циклов обучения						
	10	20	30	50	100	200	300
4	0.76	0.80	0.79	0.80	0.77	0.80	0.77
8	0.97	0.93	0.99	0.99	0.93	0.99	1
12	0.93	0.93	1	1	1	1	1
20	0.94	0.95	0.96	0.96	1	1	1
50	0.93	0.93	0.94	0.96	0.96	0.96	1

рация становится необходимой, если приложения пользователей кластера занимают не все вычислительные ресурсы узлов, на которых выполняются, в то время как более ресурсоёмкие задания вынуждены простаивать в очередях, ожидая освобождения "занятых" узлов кластера. На рис. 3, *а* приведён пример реальной задачи пользователя, которая в силу особенностей вычислительного алгоритма использует не все ресурсы узла кластера, тем не менее занимая его полностью в очереди планировщика заданий.

Системы виртуализации позволяют запускать несколько копий операционных систем (ОС) на одной машине. При этом одна из ОС выполняет служебные функции и называется гипервизором (или хостовой, от англ. host), а остальные, обращающиеся к функциям гипервизора, принято называть гостевыми. Миграция выполняющейся копии ОС доступна с использованием стандартных средств Xen.

На экспериментальном кластере ВЦ ДВО РАН [13] была добавлена возможность миграции выполняющихся процессов с одного вычислительного узла на другой, что позволяет достичь полноценной утилизации ресурсов физического вычислительного узла несколькими гостевыми ОС. Пример данных системы мониторинга при более эффективном использовании вычислительных ресурсов приведён на рис. 3, *б*.

Для тестирования Xen и его взаимодействия с Grate был создан отдельный логический кластер в составе экспериментального кластера, по пакету программ идентичный основному кластеру, но с модифицированным для поддержки Xen ядром Linux. Во время эксперимента осуществлялся процесс миграции выполняющейся копии гостевой ОС с запущенными вычислительными задачами с одного из физических узлов на другой, обладающий большими доступными ресурсами на момент миграции.

Программный комплекс на базе СМ Grate анализировал использование ресурсов на вычислительном узле, обрабатывая данные, предоставляемые промежуточным уровнем СМ (эффективная загрузка процессора, объём занятой памяти). При необходимости триггером инициировалось выполнение shell-скрипта для выполнения управляющих

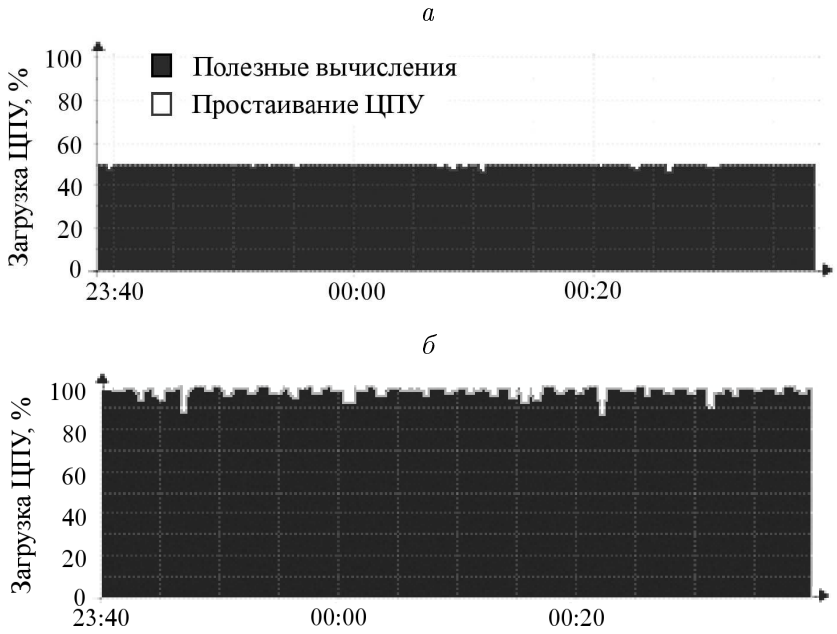

Рис. 3. Неэффективное использование вычислительного узла процессами пользователя (*а*), близкое к максимальному использование вычислительных ресурсов (*б*)

Т а б л и ц а 2. Потери производительности при миграциях гостевой операционной системы

Число миграций в час	12	6	3	2	0
Производительность, ГФлопс	8.83	9.73	10.57	10.68	11
Относительные потери производительности, %	19.72	11.55	3.9	2.9	—

команд среды виртуализации. Как показали тестовые замеры, для малого числа узлов, участвующих в расчёте параллельной задачи с интенсивной передачей данных между узлами, потери были незначительны. В табл. 2 приведены результаты для трех узлов.

Использование теста HPL позволило одновременно с проверкой корректности вычислений провести измерение производительности ВК. Следует отметить, что тест HPL достаточно интенсивно использует коммуникационную сеть. Результаты теста осреднялись на основе серии из 10 испытаний.

Для задач, в меньшей мере зависимых от интенсивности обмена данными между вычислительными узлами, потери производительности будут меньше. При отсутствии миграций использование Xen не приводит к заметным потерям производительности, что наглядно отражено в табл. 2, где 11 ГФлопс — экспериментально подтверждённая величина производительности для кластера с аналогичным набором аппаратного обеспечения [11].

Основной результат проведенного эксперимента — организация эффективного взаимодействия СМ и системы виртуализации через механизм уведомлений о событиях, сопутствующий результат — получение оценки вычислительных затрат узла при использовании на нём гипервизора Xen. В настоящее время проводятся работы по переводу части узлов гетерогенного кластера КВЦ-3 ВЦ ДВО РАН под управление Xen для проведения экспериментов с бóльшим числом многопроцессорных вычислительных узлов на задачах, менее связанных с передачей данных по коммуникационной сети. Это позволит оценить возможность использования систем виртуализации в более крупных вычислительных комплексах.

Таким образом, предложенный подход к архитектуре СМ успешно решает задачи мониторинга, осуществляя взаимодействие различных программных комплексов между собой. Механизм уведомлений о событиях позволяет улучшать функциональные возможности систем мониторинга, в том числе добавляя независимо разработанные модули, ответственные за различные аспекты анализа данных мониторинга.

Список литературы

[1] Wolski R., Spring N.T., Hayes J. The network weather service: A distributed resource performance forecasting service for metacomputing // J. of Future Generat. Comput. Systems. 1999. Vol. 15. P. 757–768.

[2] Foster I., Frey J., Tannenbaum T., Livny M., Tuecke S. Condor-G: A computation management agent for multi-institutional grids // J. on Cluster Comput. 2002. Vol. 5. P. 237–246.

[3] Genesereth M.R., Ketchpel S.P. Software Agents // Communicat. of the ACM. 1994. Vol. 37(7). P. 48–53.

[4] Ларичев О.И., Петровский А.Б. Системы поддержки принятия решений: Современное состояние и перспективы развития // Итоги науки и техники. Т. 21. М.: ВИНИТИ, 1987. С. 131–164.

[5] Ляпунов А.А. О некоторых вопросах обучения автоматов // Принципы построения самообучающихся систем. Киев, 1962. С. 115–118.

[6] Моисеев Н.Н. Математические задачи системного анализа. М.: Наука, 1981. 487 с.

[7] Яблонский С.В. Некоторые вопросы надежности и контроля управляющих систем // Математические вопросы кибернетики. Вып. 1. М.: Наука, 1988. С. 5–25.

[8] Тарасов А.Г. Расширяемая система мониторинга вычислительного кластера // Вычисл. методы и программирование. 2009. Т. 10, № 1. С. 147–158.

[9] Тарасов А.Г. Трехуровневая система мониторинга расширенной функциональности // Параллельные вычислительные технологии. Челябинск, 2008. С. 464–469.

[10] Пересветов В.В., Сапронов А.Ю., Тарасов А.Г., Шаповалов Т.С. Удаленный доступ к вычислительному кластеру ВЦ ДВО РАН // Вычисл. технологии. 2006. Т. 11. Спец. выпуск: Избранные доклады X Российской конф. “Распределенные информационно-вычислительные ресурсы” (DICR-2005). Новосибирск, 2005. С. 45–51.

[11] Пересветов В.В., Сапронов А.Ю., Тарасов А.Г. Вычислительный кластер бездисковых рабочих станций. Хабаровск, 2005 (Препр. ВЦ ДВО РАН; № 83).

[12] Писарев А.В., Тарасов А.Г. Модульная нейросетевая система мониторинга вычислительного кластера // Информационные и коммуникационные технологии в образовании и научной деятельности. Хабаровск, 2009. С. 289–296.

[13] Tarasov A.G. Integration of computing cluster monitoring system // Proc. of First Russia and Pacific Conference on Computer Technology and Applications (RPC 2010). Vladivostok: IACP FEB RAS, 2010. P. 221–224.

Поступила в редакцию 1 июля 2010 г.,
с доработки — 20 ноября 2010 г.

ЖВТ. 2012. Т. 17, № 2. С. 69–78.

Monitoring system and third-party software interaction

Tarasov A.G.

We propose an approach to the interaction of a monitoring system with a third-party software through the event notification. The practical examples of such interaction and performance, leading to estimation of the scalability of this approach are provided.

Keywords: computing clusters, monitoring systems, virtualization, artificial neural networks.