УДК 004.272.2

СОБЫТИЙНОЕ УПРАВЛЕНИЕ ОБРАБОТКОЙ ВЫХОДНЫХ ДАННЫХ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ

Леонтьев Д.В. – н.с., (ИАПУ ДВО РАН), e-mail: devozh@dvo.ru; *Одякова Д.С.* – ст. инженер-программист, (ИАПУ ДВО РАН), e-mail: darlene@dvo.ru; *Парахин Р.В.* – инженер-программист, (ИАПУ ДВО РАН), e-mail: fadak@dvo.ru; *Харитонов Д.И.* – к.т.н., с.н.с., (ИАПУ ДВО РАН), e-mail: demiurg@dvo.ru

В работе рассматривается реализация системы событийного управления обработкой данных вычислительных экспериментов. Дается пример модели вычислительного эксперимента. Дается описание архитектуры подсистемы управления. Важной чертой подхода является отсутствие необходимости в перепрограммировании исходной вычислительной задачи.

The article considers build of an event control system for processing data from computational experiments. An example of a model of a computational experiment is given. The control subsystem architecture is described. The main feature of the developed approach is that there is no need to reprogram an original computational task.

Ключевые слова: суперкомпьютерные вычисления, обработка больших данных, многопроцессорные вычислительные системы, сети Петри

Введение. С каждым годом количество экспериментов, проводящихся с помощью вычислительных кластеров, неуклонно растёт. Этому способствуют такие факторы как: дешевизна проведения экспери-

V Международная конференция «Информационные технологии и высокопроизводительные вычисления» (ITHPC-2019), Хабаровск, Россия, 16-19 сентября 2019 г. Сборник трудов. Под ред. Смагин С. И., Зацаринный А. А.

мента, скорость получения результатов, возможность проводить эксперименты параллельно и т. д.

Вычислительные кластера предназначены для одновременного выполнения множества экспериментов разных пользователей, поэтому важно использовать доступные ресурсы эффективно. В процессе проведения эксперимента возможны случаи, когда расчёт расходиться или возникают некоторые исключительные ситуации, при которых дальнейшее проведение эксперимента становиться бессмысленным. Отслеживая такие ситуации и выполняя их обработку можно увеличить эффективность использования оборудования. Отслеживать можно и другие события, такие как запись в файл, запрос данных, запись в базу данных, взаимодействие с сокетами и т. д. Имея информацию о происходящих событиях появляется возможность строить пользовательские схемы обработки данных вычислительных экспериментов. Примером может служить визуализация данных эксперимента. Визуализация данных может требовать значительных вычислительных ресурсов и дискового пространства, поэтому наиболее выгодным вариантом становится визуализация данных непосредственно на кластере [1, 2], которая происходит параллельно процессу выполнения эксперимента. Такой вариант позволяет сократить время ожидания получения результатов и избавиться от загрузки результатов «неудачных» экспериментов на компьютер пользователя, что существенно экономит время. Дополнительным плюсом при визуализации данных является разгрузка сетевых соединений и как следствие более быструю доставку результатов до конечного потребителя. Так в процессе визуализации данные, занимающие гигабайты дискового пространства могут быть преобразованы к читаемому пользователем виду, занимающему в сотни, а то и в тысячи раз меньше места.

В настоящей работе рассматриваются реализация событийного управления обработкой данных вычислительных экспериментов.

Модель обработки данных. В данной работе в качестве примера вычислительного эксперимента используется моделирование волн цунами. Вычислительный эксперимент носит итерационный характер. Итерации происходят по расчётному времени. Итерации отличаются друг от друга по значению этого времени. Раз в заданный интервал времени происходит запись расчётных массивов в файл с уникальным именем (например, eta0001, eta0002 и т.д.). Раз в заданный интервал модельного времени происходит запись усреднённых величин в лог. После записи усреднённых величин необходимо визуализировать рас-

чётные массивы (данные берутся из файла, записанного до записи лога). По окончанию расчета массивы данных сохраняются в файлы, и записывается информация в лог. После этого происходит визуализация всех записанных массивов и склейка в видеоролик.

Модель данного вычислительного эксперимента представлена на рис. 1. Модель представляет собой сеть Петри, состоящую из множества мест, множества переходов, входной функция инцидентности переходов и выходной функции инцидентности переходов. Процесс построения модели вычислительного эксперимента рассмотрен в статье [3].

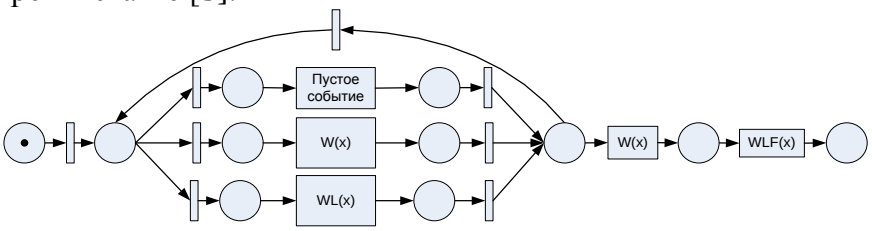

Рис. 1 Модель вычислительного эксперимента

Модель управляющего процесса показана на рис. 2. В данном примере используется шаблон «реакция на предыдущее событие».

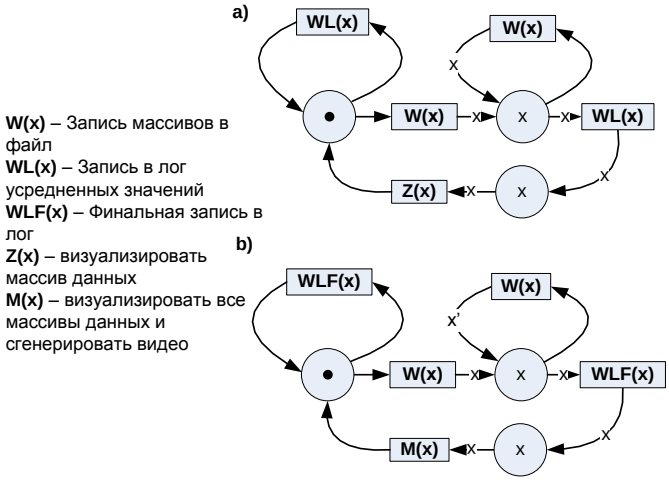

Рис. 2 Модель управляющего процесса: a) реакция на событие запись усредненных значений в лог, b) – реакция на событие финальная запись в лог.

На рис. 2(a) приведена модель реакции на событие запись усредненных значений в лог, происходящая в вычислительном эксперименте в цикле. Т.е. в процессе вычислительного эксперимента происходит получение события W(x). Параметры данного события сохраняются и передаются далее. Получение события W(x) происходит до тех пор, пока не будет

получено событие WL(x), после которого будет выполнена процедура визуализации последнего массива данных Z(x), полученного в событие W(x). А на рис. 2(b) представлена реакция на событие финальной записи в лог. Отличие данной реакции от предыдущей заключается в отслеживаемом событии WLF(x) (финальная запись в лог) и реакции на него M(x) (визуализация всех массивов данных и генерация видео).

Архитектура подсистемы управления. Отслеживание событий вычислительного процесса является малозатратной деятельностью и не потребляет значительного количества ресурсов, в то время как реакция на события может быть длительным и ресурсоёмким процессом. В рамках данной статьи обработка данных в ходе реакции рассматривается как самостоятельный процесс, который выполняется параллельно основному на выделенном узле (узел обработки).

Архитектура подсистемы управления представлена на рис. 3. Белые прямоугольники представляют собой типы узлов. Прямоугольники темно серого цвета обозначают процессы. Прямоугольники светло серого цвета обозначают действия. Прямоугольник со скругленными краями обозначает задание. Стрелками показано передача команд и данных. Пунктирными стрелками изображаются действия, которые выполняются в процессе выполнения скриптов.

В общем виде процесс запуска задания с событийным управлением выглядит следующим образом. Сначала пользователь строит модель вычислительного эксперимента. Далее формируется задание для системы очередей. Выполнение задания начинается со скрипта Prolog на управляющем узле. Этот скрипт передает данные вычислительного эксперимента (модель эксперимента, параметры задания и т.д.) на узел обработки, на котором создается экземпляр управляющего процесса. Управляющий процесс имеет информацию о выполняющейся задаче (номер задачи, какие узлы используются и д.т.) и получает события от процесса «Auditd Control». После выполнения скрипта Prolog на управляющем узле задача поступает на вычислительный узел. Перед выполнением задачи на вычислительном узле происходит запуск prolog скрипта, который добавляет правила отслеживания событий в Auditd, определенные при построении модели вычислительного эксперимента. После окончания выполнения задачи происходит удаление правил отслеживания событий из Auditd. Auditd отслеживает события согласно определенным правилам и передает «пойманые» события на узел обработки, где происходит отслеживание событий и выполнение реакций на эти события.

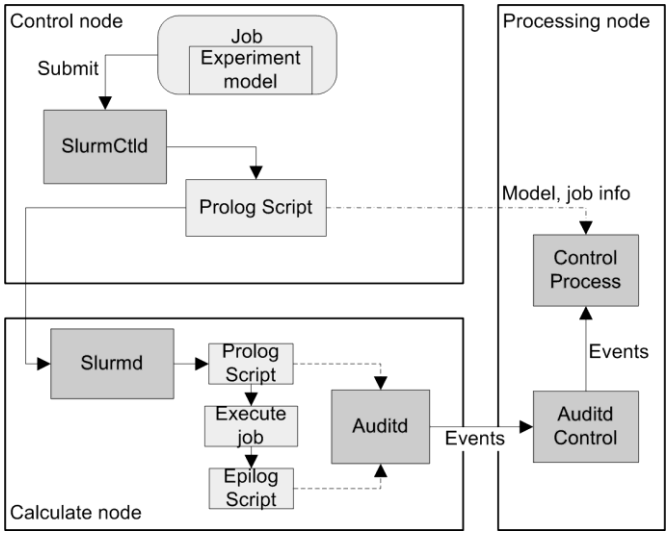

Рис. 3 Архитектура подсистемы управления

Перехват событий без изменения кода программы. На вычислительных кластерах используется различное программное обеспечение. Это могут быть как специализированные пакеты, так и программные коды разработанные пользователями самостоятельно. Изменение исходных кодов, с целью контроля событий, достаточно трудоемкий процесс и не всегда выполнимый, так как программные пакеты могут поставляться в скомпилированном виде или может статься так, что исходные коды недоступны или утеряны.

Другим способом получения информации о происходящих в программе событиях является отслеживание событий операционной системы. Для этого применяется Audit – подсистема ядра linux. Audit позволяет отслеживать события операционной системы по заданным правилам.

После построения модели вычислительного эксперимента пользователь формирует правила для отслеживания событий. В данной работе допустимы только следующие события: запись в файл, запись в базу данных, запрос данных, Fail, взаимодействие с сокетами, запуск программы. Для формирования правила для каждого типа события используется определенный набор параметров. Например, правило для события запись в файл выглядит следующим образом:

auditctl -w /dir/file –p w -k flag, где /dir/file – наблюдаемый файл,
flag – флаг, который добавляется к событию.

Так выглядит правило, которое добавляется в демон Auditd, расположенный на вычислительном узле.

Заключение. В статье рассмотрен принцип работы событийного управления обработкой данных вычислительных экспериментов. Перехват событий осуществляется с помощью подсистемы Audit. Альтернативным средством для отслеживания событий может выступать inotify, однако, inofity позволяет отслеживать только события файловой системы. Поэтому inotify не подходит для данной задачи. Главным особенностью разработанного решения является отсутствие необходимости в перепрограммировании существующих программных средств. Подсистема событийного управления обработкой данных вычислительных экспериментов реализована в рамках системы управления прохождения заданиями WBS [4]. В дальнейшем авторами планируется разработка удобного пользовательского WEB интерфейса, с помощью которого пользователи смогут использовать возможности системы.

Благодарности. Исследования выполнены при поддержке комплексной программы фундаментальных исследований ДВО РАН «Дальний Восток», проект 18-5-078.

Библиографические ссылки

1. Джосан О.В. О визуализации научных данных для высокопроизводительных параллельных приложений // Тезисы конференции ПАВТ 2009. Н. Новгород, 2009.
2. Корж О.В. Автоматическое построение передаточных функций для систем визуализации распределенных данных на суперкомпьютерах // Параллельные вычислительные технологии (ПАВТ 2012): тр. Междунар. науч. конф. (Новосибирск, 26-30 марта 2012). Челябинск: ЮУрГУ, 2012. С. 726.
3. Leontyev D.V., Kharitonov D.I., Tarasov G.V. The data processing model of computational experiments // 20th Conference Scientific Services and Internet, SSI 2018, CEUR-WS. Vol. 2260. P. 373-386.
4. Одякова Д.С., Тарасов Г.В., Харитонов Д.И. Система WBS как расширенный инструмент управления вычислительной средой // Суперкомпьютерный форум «Суперкомпьютерные технологии в образовании, науке и промышленности», 26-28 ноября 2012 г. Н. Новгород: Нижегород. гос. ун-т им. Н.И.Лобачевского, 2012.