РАСПРЕДЕЛЕННАЯ СИСТЕМА СБОРА И ОБРАБОТКИ ИНФОРМАЦИИ О СОСТОЯНИИ ПРОГРАММНО-АППАРАТНОЙ СРЕДЫ СУПЕРКОМПЬЮТЕРОВ

К.С. Стефанов

1. Введение

Архитектуры современных суперкомпьютеров становятся все сложнее, количество однотипных элементов (вычислительных узлов, процессоров, ядер) растет, увеличивается и разнообразие типов элементов, появляются различные ускорители, системы все больше становятся гетерогенными. Написание эффективного программного обеспечения для таких систем становится все более трудной задачей. Причин низкой эффективности – целое множество, выделить что-то одно практически невозможно. Для контроля процесса выполнения параллельных программ нужно отслеживать десятки параметров на тысячах, а в перспективе и на сотнях тысяч и миллионах вычислительных узлов. В таких условиях традиционные системы сбора и обработки информации о состоянии программно-аппаратной среды (системы мониторинга) перестают работать, поэтому и возникает необходимость в разработке специализированных систем.

С другой стороны, увеличение размеров вычислительных систем делает все более частыми единичные отказы отдельных компонентов, которые тоже нужно отслеживать. При этом количество параметров, сигнализирующих о «здоровье» вычислительной системы, также исчисляется десятками: скорости вращения вентиляторов, температуры составных частей вычислительных узлов, наличие ошибок памяти, дисков, параметры работы систем охлаждения, электропитания и т.п. Важно и то, что это не только «здоровье» системы, но и эффективность работы приложений: ошибки InifinBand и Ethernet, ошибки дисков, ошибки ECC, понижение частоты процессора из-за перегрева – все это ведет к снижению реальной производительности.

В табл. 1 приведены некоторые цели мониторинга суперкомпьютеров.

Таблица 1. Цели мониторинга суперкомпьютеров

Цель мониторинга	**Требуемая надежность**	**Сложность обработки**	**Объем данных**
Аварийные ситуации	Очень высокая	Низкая	Небольшой
Работоспособность узлов	Средняя	Средняя	Средний
Функциональность узлов и вычислительной инфраструктуры	Средняя	Высокая	Средний
Мониторинг эффективности	Низкая	Низкая	Высокий

Отслеживание аварийных ситуаций (проблемы с охлаждением, электропитанием и т.п.) требует очень высокой надежности, так как ценой пропуска событий, сигнализирующих о таких ситуациях, может стать пожар, выход из строя дорогостоящего оборудования суперкомпьютеров, потеря данных пользователей и т.п. Поэтому требуется дублирование аппаратуры и ПО, осуществляющего этот мониторинг. При этом объем данных, которые приходится анализировать, невелик, сложность их анализа также небольшая (обычно это сравнение с заданным порогом). С такими задачами справляются и существующие системы мониторинга, требуется лишь их настройка на оборудование и задание необходимых реакций на нештатные ситуации.

Под работоспособностью вычислительных узлов мы понимаем исправность тех компонентов, которые отвечают за физическое поддержание минимальной работоспособности: блоки питания, вентиляторы, отсутствие перегрева и т.п. Пропуск событий, связанных с выходом из строя этих компонентов, чреват выходом из строя соответствующих вычислительных узлов. В то же время современные компоненты суперкомпьютеров, как правило, имеют встроенную защиту от таких аварий. Поэтому требования по надежности отслеживания событий такого рода не столь велики, как для мониторинга инфраструктуры суперкомпьютера. Поток данных для такого мониторинга растет пропорционально числу вычислительных узлов, поэтому он достаточно велик для современных больших вычислительных систем. Сложность обработки этих данных больше, чем для отслеживания аварий общей инфраструктуры суперкомпьютера. Существующие системы мониторинга в состоянии справиться с отслеживанием таких аварийных ситуаций, но надо учитывать объем данных, который должен проходить через систему мониторинга.

Под мониторингом функциональности узлов и вычислительной инфраструктуры мы понимаем отслеживание пригодности аппаратуры и программного обеспечения для выполнения основного назначения суперкомпьютеров — эффективного выполнения задач пользователей. Для этого надо отслеживать состояние коррекции ошибок (ECC) оперативной памяти, отсутствие сбойных секторов на жесткий дисках (при помощи SMART или же отслеживая время выполнения соответствующих операций), ошибок на сетевых интерфейсах, правильность работ механизмов автоматического определения скоростей сетевых интерфейсов и т.п. Наличие ошибок такого рода может привести либо к сильному замедлению работы пользовательских задач, либо к их аварийному завершению и необходимости повторного запуска, в итоге это выливается в неэффективное

использование имеющихся ресурсов. Поток данных, необходимый для такого вида мониторинга, больше, чем для мониторинга работоспособности узлов, так как больше число отслеживаемых параметров. Алгоритмы анализа также сложнее, так как для большинства параметров требуется отличать единичные ошибки, которые могут возникать в процессе нормальной работы, от уровня ошибок, свидетельствующего о наличии проблемы. Кроме того, для некоторых параметров требуется отслеживать их взаимосвязь (например параметры сетевых интерфейсов требуется отслеживать с обоих концов канала связи). Однако уровень требуемой надежности здесь еще ниже, чем для отслеживания работоспособности, так как пропущенные проблемы такого рода хотя и вызовут снижение эффективности использования имеющихся ресурсов и, возможно, потребуют повторный расчет какой-то части задач, тем не менее не приведут к физической порче оборудования.

Мониторинг эффективности программ [1, 2] – это исследование свойств программ и поиск возможных причин их неэффективного поведения по данным о состоянии программно-аппаратной среды суперкомпьютера. Поток информации, который необходимо анализировать в рамках этого подхода, очень велик: от десятков до сотен параметров для каждого вычислительного узла (а значительная часть — для каждого процессорного ядра), снимаемых с периодичностью в единицы секунд, а в перспективе и с большей частотой. Обработка этих данных, выполняемая одновременно с их получением, довольно проста: прореживание путем удаления мало изменяющихся значений, сравнение с порогом, сохранение в базу данных для дальнейшего анализа. Еще раз подчеркнем, что простые алгоритмы используются лишь в режиме реального времени, до сохранения полученных данных. Алгоритмы, осуществляющие последующий анализ, имеют гораздо большую сложность, но на них накладываются гораздо менее жесткие ограничения по времени работы. При этом требования по надежности систем, осуществляющих сбор и обработку таких данных, не предполагают каких-то специальных мер по повышению отказоустойчивости. Прекращение сбора и анализа данных не приведет ни к порче оборудования, ни к простоям, ни к необходимости перезапуска каких-то задач, за исключением тех, для исследования которых используется этот сервис.

Таким образом, существует несколько разных задач, требующих обработки данных мониторинга. Эти задачи различаются по объему данных, необходимому для их решения, по сложности алгоритмов анализа этих данных, и по предъявляемым требованиям к надежности (отказоустойчивости) подсистем, решающих каждую из этих задач.

Однако существующие системы мониторинга (Zabbix, Ganglia, Nagios и т.п.) не позволяют свести решение всех этих задач в рамках единого комплекса. Несмотря на то, что часть из указанных задач может быть решена в рамках существующих систем мониторинга, их объединение в рамках одного установленного экземпляра системы мониторинга порождает некоторые трудности.

Вся обработка данных мониторинга в существующих системах осуществляется на выделенных узлах, не являющихся вычислительными узлами. С одной стороны, такая архитектура снижает нагрузку на вычислительные узлы, уменьшая влияние системы мониторинга на выполняющиеся на них задачи. С другой стороны, вычислительные ресурсы, необходимые даже для простейшей обработки потоков данных, если они генерируются каждым процессорным ядром большого суперкомпьютера с периодом в единицы секунд, весьма значительны. Перенос части такой обработки непосредственно на вычислительные узлы заметно не увеличит нагрузку на них, но в то же время позволит в сумме добавить значительные ресурсы для обработки данных мониторинга.

Существующие системы мониторинга предполагают либо дублирование обработки данных, либо выстраивание иерархической структуры обработки. При этом гибкости для выделения путей прохождения отдельных потоков данных не предусматривается. На практике это означает, что если мы хотим обеспечить дублирование при обработке какой-то части данных для обеспечения требуемой надежности, нам приходится дублировать обработку всех потоков данных, что приводит к выделению дополнительных ресурсов для надежной обработки и сохранения в том числе и тех потоков данных, для которых обеспечение отказоустойчивости не требуется.

Кроме того, реализованный в существующих системах мониторинга набор алгоритмов обработки данных довольно небогат, и реализация новых непроста, поэтому использование существующих систем в исследованиях новых способов анализа данных мониторинга сталкивается с трудностями. Также затруднено программное изменение настроек системы мониторинга, например изменение частоты опроса датчиков в моменты простоя отдельных вычислительных узлов, что поможет уменьшить необходимые ресурсы для хранения собранных данных.

2. Предлагаемый подход

Для преодоления указанных недостатков предлагается создание новой системы мониторинга. Далее будет описаны принципы построения новой системы и ее предполагаемая архитектура.

Система мониторинга является распределенной и состоит из агентов мониторинга. Агенты мониторинга осуществляют сбор данных и их обработку, а также могут передавать данные другим агентам для дальнейшей обработки.

Агент мониторинга строится по модульному принципу. Все получение и обработка информации о состоянии вычислительной системы ведется в узлах. Узел, который получает данные мониторинга от

программной-аппаратной среды, называется датчиком. Данные передаются от узла к узлу. При этом каждый узел конфигурируется независимо от других. Так же независимо создаются связи между узлами.

В системе передаются сообщения двух типов. Основной тип — сообщения, содержащие данные мониторинга. Они передаются от узла к узлу вдоль созданных связей между узлами. Второй тип — управляющие сообщения, они передаются непосредственно от одного узла к другому вне зависимости от наличия между ними связей.

Система исполнения — это программное окружение, которое обеспечивает функционирование узлов, связей между ними и передачу сообщений.

Опишем подробнее компоненты создаваемой системы.

Каждый узел имеет определенный тип. Тип узла задается при его создании, и определяет функции, которые вызываются системой исполнения, когда узлу надо выполнить какое-то действие. Количество узлов каждого типа не ограничено.

Общий вид узла мониторинга представлен на рис. 1.

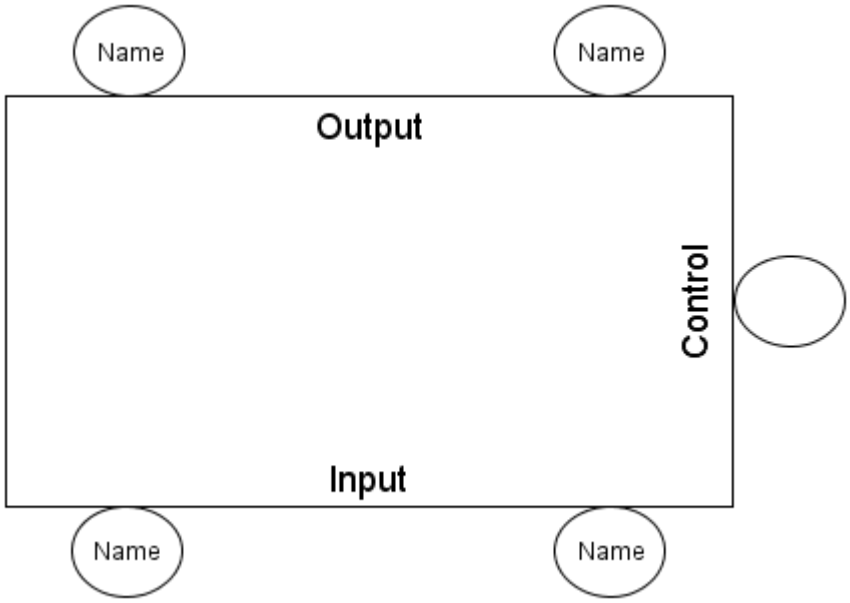

Рис.1. Общий вид узла мониторинга

Каждый узел имеет ноль или больше именованных входов (Input), ноль или больше именованных выходов (Output), и может принимать и передавать управляющие сообщения (Control). Все узлы имеют числовой идентификатор, присваиваемый системой исполнения при их создании. Кроме того, узлу может быть присвоено имя (строка символов).

При создании связи между узлами задаются узел-источник данных (при помощи имени или идентификатора), имя выхода этого узла, узел-приемник данных и имя входа, на который данные будут передаваться. После создания связи копия данных, которые узел будет передавать на данный выход, будут передаваться системой исполнения на все входы, связанные с данным выходом.

Формат данных, передаваемых между узлами, фиксирован. Это последовательность четверок (Id, Index, Len, Value), где Id – целое число, определяющее тип данных, Index – идентификатор объекта, к которому относятся данные, Len – длина данных (поля Value) в байтах, и Value – сами данные. Интерпретация содержимого полей Value и Index осуществляется узлами, обрабатывающими данные. Index может задавать, например, номер процессорного ядра или номер сетевого интерфейса, к которому относятся данные мониторинга.

Управляющие сообщения служат для создания и уничтожения узлов, задания их имени, создания и уничтожения связей между узлами, создания таймеров и подписки на них (см. ниже) — такие сообщения обрабатываются системой исполнения. Сообщения, служащие для изменения настроек узлов и другие, специфичные для конкретного узла (типа узлов), обрабатываются непосредственно узлами, которым они адресованы. Управляющее сообщение может быть послано любым узлом любому другому. Для определения получателя сообщения используется имя или номер узла-получателя.

Еще один элемент, необходимый для работы системы мониторинга — таймеры. При создании таймера определяется имя таймера, а также в какие моменты времени он будет срабатывать. Это может быть один предопределенный момент времени (абсолютный или относительно текущего времени), или таймер может срабатывать с заданной периодичностью. При срабатывании таймер посылает управляющее сообщение всем

подписанным на него узлам. Для подписки используется управляющее сообщение. Узел может подписать на таймер самого себя или другой узел.

Все взаимодействие внутри системы реализовано в функциональном стиле. Узел определяет функции, которые вызываются для обработки получения данных мониторинга, получения управляющих сообщений, при инициализации и уничтожения самого узла, создании связи и т.п.

Для построения системы мониторинга используются несколько видов узлов. Мы говорим о нескольких видах, но они отличаются лишь с точки зрения их функциональности, системой исполнения все узлы обрабатываются одинаково.

Датчик — это узел без входов. Его назначение — по сигналу от таймера получить данные мониторинга путем обращения к аппаратуре, сервисам ОС и т.п., и передать его на свой выход (у датчиков он обычно один).

Узел со входами и выходами обычно осуществляет обработку данных мониторинга. Обработка может быть любой: фильтрация данных, вычисление скорости изменения величины (как например вычисление скорости передачи и приема данных по счетчикам переданных и полученных байт), сравнение с заданным порогом и т.д. Именование входов и выходов нужны для разделения потоков данных. Например, узел, отбирающий данные по какому-то критерию, может передавать данные, подходящие под критерий фильтрации, на выход с именем «match», а данные, не подходящие под критерий — на выход с именем «notmatch». Соединив эти выходы с другими узлами, мы получаем разделение потоков данных, которые затем могут обрабатываться независимо.

Узлы со входами, но без выходов предназначены для передачи данных мониторинга вовне агента. Чаще всего такой узел будет использоваться для организации взаимодействия агентов. Узел принимает данные мониторинга на своих входах, сериализует их и передает по сети другому агенту. На входе принимающего агента будет работать узел-приемник, который во многом устроен как датчик (тоже не имеет входов), однако данные он получает не от аппаратуры или ОС, а принимает их по сети от другого агента. Настройка приемников и передатчиков (куда передавать данные, на каком сетевом адресе принимать соединения, по какому протоколу производить обмен и т.п.) осуществляется путем посылки управляющих сообщений.

Другими вариантами узлов без выходов могут быть узлы, сохраняющие данные мониторинга во внешнюю базу данных или запускающие внешние реакции: отсылающие сообщение администратору по электронной почте, SMS, записывающие сообщение в журнал или же инициирующие выключение каких-то компонентов вычислительной системы.

Так как каждый датчик получает сообщение от таймера независимо от других и отсылает полученные им данные мониторинга в отдельном сообщении, то для эффективности необходимо собирать данные из нескольких таких сообщений в одно прежде чем передавать их по сети. Такой сборкой занимается узел консолидации. Этот узел при получении любого сообщения с данными мониторинга сохраняет это сообщение у себя во внутреннем буфере и делает системе исполнения специальный запрос (специальное управляющее сообщение). По этому запросу система исполнения создает список всех таймеров, которые активны в момент запроса (рассылают сообщения подписанным на них узлам), и посылает узлу консолидации управляющее сообщение после того, как все активные таймеры закончат оповещение о своем срабатывании. До получения этого сообщения узел консолидации запоминает в буфере все получаемые им сообщения с данными мониторинга. К моменту получения сообщения об окончании срабатывания таймеров все датчики, которые должны были сгенерировать данные по сигналу активных таймеров, уже закончили это делать, и узел консолидации собирает накопленные данные в одно сообщение, а затем передает его дальше для обработки (например, для передачи по сети).

В момент старта система исполнения читает из конфигурационного файла список всех типов узлов, которые будут доступны в процессе работы, и инициализирует соответствующие модули. Модуль представляет собой динамически подключаемую библиотеку, в которой реализованы все функции, необходимые для работы узлов типов, реализованных в модуле (один модуль может реализовывать несколько типов). Затем система читает из конфигурационного файла тип узла, который надо создать первым (стартовый узел). Этот узел создается и получает управляющее сообщение, в котором содержится номер файлового дескриптора, указывающего на оставшуюся часть конфигурационного файла. В дальнейшем стартовый узел читает оставшуюся часть конфигурационного файла и интерпретирует его по своему усмотрению. В частности, там могут содержаться команды для создания других узлов, связей между ними, таймеров, подписки на них узлов и задания прочей конфигурации. После окончания обработки стартовым узлом посланного ему управляющего сообщения, система исполнения переходит в нормальный режим работы. В дальнейшем изменение конфигурации может производиться любым узлом. Это может быть реакцией на какие-то события, обнаруженные в данных мониторинга, либо же узел может принимать сетевые соединения и реагировать на полученные по сети команды. Эта функциональность не заложена в систему исполнения, и должна реализовываться в узлах.

3. Пример построения системы мониторинга

Для примера приведем строение агента мониторинга, по функциям соответствующее агентам мониторинга «классических» систем. Этот агент состоит из двух датчиков, которые по сигналу таймера

собирают данные от аппаратуры и передают их по сети для обработки в другие части системы мониторинга. Строение такого агента показано на рис. 2.

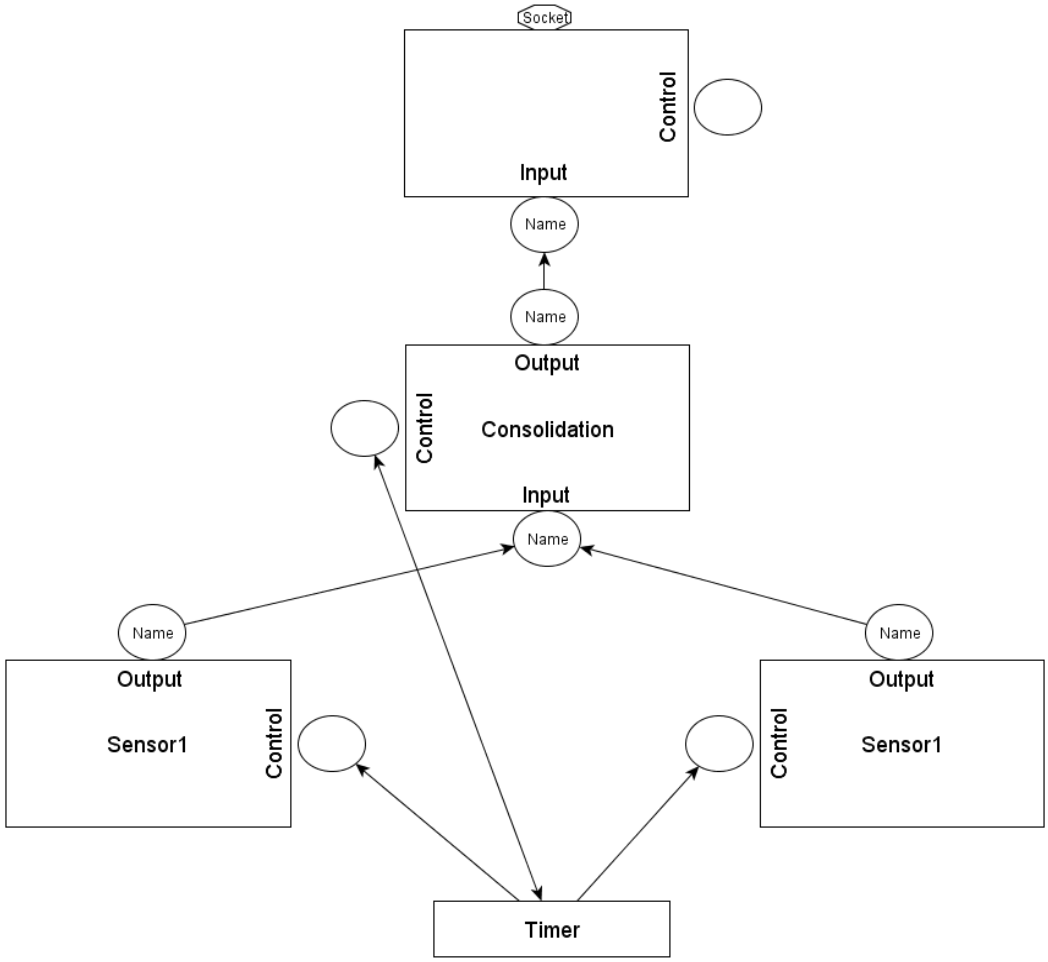

Рис. 2. Агент мониторинга с двумя датчиками

Для упрощения на рисунке не показан стартовый узел, который при запуске создает все нужные узлы и осуществляет настройку.

Узлы «Sensor1» и «Sensor2» — датчики. Например, это могут быть датчики загрузки процессора и датчики загрузки сетевых интерфейсов. Таймер с заданной периодичностью активирует датчики, они собирают данные и предают их узлу консолидации. Узел консолидации накапливает данные от обоих датчиков и передает их узлу-передатчику для передачи на другие агенты для обработки.

4. Текущее состояние работы

В настоящее время идут работы по реализации системы исполнения. Реализованы механизмы создания и уничтожения узлов, создания связей между ними, передачи данных мониторинга. До конца 2013 г. планируется в основном завершить реализацию системы исполнения и провести эксперименты на реальных вычислительных системах для проверки предложенной концепции.

5. Заключение

В работе предложена новая концепция и основанная на ней архитектура системы мониторинга для суперкомпьютеров. Предложенная концепция обладает гибкостью и позволит решать в рамках единого подхода задачи, связанные с обработкой данных о состоянии программно-аппаратной среды суперкомпьютеров.

Работа выполняется при финансовой поддержке РФФИ, грант № 13-07-00775 А

ЛИТЕРАТУРА:

1. Никитенко Д.А., Стефанов К.С. Исследование эффективности параллельных программ по данным мониторинга // Вычислительные методы и программирование.- 2012.- Т. 13.- Раздел 2. С. 97-102 (http://num-meth.srcc.msu.ru/)
2. Воеводин Вад.В., Стефанов К.С., Никитенко Д.А., Адинец А.В., Брызгалов П.А., Жуматий С.А. Job Digest - подход к исследованию динамических свойств задач на суперкомпьютерных системах // Труды

Международной суперкомпьютерной конференции "Научный сервис в сети Интернет: поиск новых решений" (17-22 сентября 2012 г., г. Новороссийск).- М.: Изд-во МГУ, 2012.- С. 9.