УДК 004.021

КОМПЛЕКСНЫЙ АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ, ОСНОВАННЫЙ НА ДАННЫХ СИСТЕМНОГО МОНИТОРИНГА

Д. А. Никитенко[1]

Всестороннее исследование эффективности использования суперкомпьютерных ресурсов представляет собой задачу исключительно важную, с которой сталкивается каждый владелец и каждый пользователь такого рода системы. В статье описывается подход к комплексной оценке производительности, включающей в себя специфику выполнения отдельных суперкомпьютерных приложений, распределение задач по очередям и суммарное использование ресурсов системы. Подход основан на анализе данных системного мониторинга и ориентирован на предоставление возможности качественной оценки поведения суперкомпьютерных приложений и систем в целом.

Ключевые слова: суперкомпьютер, производительность, эффективность, параллельные программы, динамические характеристики, загрузка, мониторинг.

1. Введение. Суперкомпьютерные системы отличаются от менее производительных вычислительных машин практически всем: масштабом, числом компонентов и их разнообразием, сложностью архитектуры, энергопотреблением, большей требовательностью к развитости инфраструктуры в целом и др. [1, 2]. В связи с этим уже с этапа разработки высокопроизводительной вычислительной системы необходимо намного более глубокое понимание динамики происходящего вокруг нее. Для сравнения можно привести обычный городской автомобиль, массовый, как аналог обычного компьютера, и в противоположность ему, по аналогии с суперкомпьютерной системой, автомобиль спортивный. В обоих случаях архитектура должна соответствовать решаемым задачам, в обоих случаях внутреннее устройство весьма сложно, но жесткость требований к эффективности существенно отличается. Естественно, для контроля над происходящим в простом автомобиле хватит нескольких десятков базовых характеристик, из которых водителю (пользователю) вообще будут показаны лишь основные. В случае же спортивного автомобиля ведется мониторинг существенно большего числа характеристик, многие из которых необходимо получать с более высокой точностью и детализацией. Львиная доля собранных данных анализируется не только в автоматическом режиме, но и анализируется целой командой экспертов. При этом в случае суперкомпьютеров разница в количестве характеристик, которые требуется отслеживать, еще более велика, так как одним из важнейших отличий таких систем от рядовых ПК и серверов является колоссальная разница в масштабах [3]. Принимая во внимание стоимость суперкомпьютерных систем и размера эксплуатационных расходов, требования к полноте и точности сведений о состоянии системы, ее работоспособности и эффективности использования крайне высоки.

Значительная часть компонентов аппаратной составляющей суперкомпьютера [4] обладает заложенным производителем набором специальных характеристик, значения которых можно некоторым образом получать. Изначально бóльшая часть таких характеристик была реализована в виде так называемых датчиков для собственно отладки компонента производителем. Однако та часть характеристик, которая остается доступной пользователю, может быть успешно использована для оценки не столько эффективности работы самого компонента, сколько для оценки поведения всей связки ПО и аппаратуры в конкретных условиях.

В результате на каждом узле доступен достаточно большой набор, насчитывающий десятки характеристик, описывающих текущее состояние компонентов программно-аппаратного комплекса: центрального процессора, сетевых интерфейсов, системы ввода-вывода, операционной системы и др. Использование такого рода данных, собранных с помощью некоторой системы мониторинга, дает возможность получить своеобразный профиль активности вычислительной системы, “увидеть”, что происходило с каждой ее составляющей, а также со всей системой или некоторым множеством узлов. Если же рассмотреть только то множество узлов, на котором выполнялась отдельная задача, то будет получен профиль этой задачи.

[1] Научно-исследовательский вычислительный центр, Московский государственный университет им. М. В. Ломоносова, Ленинские горы, 119992, Москва; науч. сотр., e-mail: dan@parallel.ru

Идея системного мониторинга состоит в следующем. Выделяется (и обычно фиксируется) некоторый набор датчиков — динамических характеристик состояния программной среды и аппаратуры, которые могут принимать определенные значения и считываться с определенной же частотой. Каждый датчик имеет уникальный идентификатор, возможность получить свое значение от операционной системы, переменных окружения или же от доступных интерфейсов аппаратуры через программу-агент. Система мониторинга периодически получает значения всех датчиков. После этого данные или обрабатываются "на лету", или сохраняются для последующего анализа. Таким образом получается последовательность мгновенных состояний вычислительной системы. Следует отметить, что анализ данных системного мониторинга позволяет строить лишь гипотезы, помогать выявлять аномалии и фрагменты необычного поведения приложения, но не может давать ответ на вопрос, где конкретно в программе имеется ошибка или источник неэффективности.

Основные подходы к исследованию эффективности приложений и вычислительных систем, основанные на анализе данных системного мониторинга, отличаются рядом ключевых моментов: привязанностью к некоторой системе мониторинга или способу хранения данных, числом и составом рассматриваемых характеристик, частотой их сбора, возможностями анализа. Отсутствие необходимости внесения изменения в текст программы, а значит, и широкая доступность подхода для специалистов всех уровней подготовки, включая начальный, — вот наиболее сильные аргументы в пользу подходов, основанных на данных системного мониторинга.

2. Причины и признаки снижения производительности. Предлагаемый в настоящей статье подход, реализующий комплексный анализ производительности и эффективности суперкомпьютерных систем и приложений, прежде всего ориентирован на то, чтобы помочь исследователю ответить на вопрос о возможной локализации и природе основных причин, негативно влияющих на процесс выполнения задач.

Здесь особенно важно подчеркнуть, что причины снижения эффективности могут составлять сложную иерархию, в которой истинные причины находятся на самом нижнем слое. Наибольший эффект от оптимизации будет достигнут при устранении именно этих причин. Настоящие причины снижения эффективности могут скрываться в нехватке ресурсов аппаратуры или ее некорректной работе, будь то вследствие неподобающей настройки или же как следствие выхода из строя каких-то компонентов. Кроме того, они могут скрываться в логике распределения задач по узлам, в особенностях настройки системного ПО, не говоря уже о массе особенностей реализации пользователем приложения, включая компиляцию и параметры запуска в условиях конкретных входных данных и состояния окружения.

Рассмотрим пример, иллюстрирующий то, что мы понимаем под причинами и признаками снижения эффективности. Допустим, в ходе анализа динамики поведения задачи выясняется, что бóльшую долю времени выполнения задача провела на реализации операций ввода-вывода. При этом реальной причиной снижения эффективности (а ввод-вывод — одна из самых затратных операций) является не столько сам факт значительного пребывания в состоянии ввода-вывода, сколько то, что это состояние вызвало. Факт длительного пребывания в состоянии ввода-вывода является признаком неэффективного поведения программы. Причиной интенсивного ввода-вывода может оказаться неустранимая особенность алгоритма или программной реализации, но в других случаях проблема может решаться просто путем изменения каких-то настроек, скажем, частоты сохранения врéменных данных программы. Кроме того, может влиять недостаточный объем доступной памяти или другие факторы.

Задача сопоставления признаков (некоторого характерного поведения одной или нескольких динамических характеристик) и реальных причин снижения производительности является крайне важной для подходов, основанных на системном мониторинге. Об этом направлении исследования будет еще сказано далее.

3. Сбор и хранение данных системного мониторинга. При всех преимуществах подходов, основанных на данных системного мониторинга [5], для них существуют критически важные моменты, определяющие ограничения в возможностях проведения анализа.

Первым и наиболее существенным элементом в цепочке является непосредственно получение данных, т.е. система мониторинга. При росте масштабов вычислительной системы и увеличении частоты съема данных увеличивается поток данных, что выливается в существенную проблему для систем высокого уровня производительности. Во многом проблема состоит в том, что большинство систем мониторинга не были рассчитаны на эксплуатацию в условиях больших систем, большого числа метрик и высокой частоты съема данных. При разработке бóльшая их часть была ориентирована на периодическую проверку отдельных характеристик. Недостаточные возможности доступной системы мониторинга приводят к сокращению числа исследуемых характеристик и более грубой грануляции, что в некоторых случаях может скрывать нюансы поведения исследуемого объекта. Однако и избыточно высокая частота данных может

усложнить процесс анализа (особенно для таких характеристик, как количество операций с плавающей точкой), сделав необходимой процедуру сглаживания для визуального восприятия.

Вторым важнейшим вопросом является вопрос сохранения данных мониторинга для апостериорного анализа. Существуют средства хранения, ориентированные на потоковое сохранение данных достаточно большого масштаба, однако именно такая ориентированность становится проблемой при попытке последующего доступа к данным. Из этих соображений представляется более целесообразной обработка и агрегация данных "на лету". Это позволит не просто сохранять меньшие объемы данных, но и избежать избыточной обработки данных. Сырые данные системного мониторинга могут требовать значительных объемов, а потому ресурсов для хранения и обработки. Сделаем грубую прикидку масштабов для простого примера. Система "Ломоносов" Суперкомпьютерного комплекса МГУ [6] — это около 6000 вычислительных узлов. В данный момент ведется мониторинг, позволяющий снимать около 50 характеристик с частотой порядка 1 Гц. Это соответствует потоку приблизительно 5 кбайт/с с узла. Если рассмотреть данные в том виде, в котором они должны быть сохранены с учетом возможности дальнейшего формирования выборок по ним (индексы и т.п.), то это уже соответствует потоку с узла примерно 25 кбайт/с. Таким образом, со всей системы поток для сохранения составил бы 150 Мбайт/с, что находится на грани возможной скорости линейной записи современными жесткими дисками. При попытке чтения такая скорость работы с диском уже станет недостижима, а значит, требуются другие, существенно более дорогие средства хранения. С точки зрения суммарных объемов, такой поток выливается приблизительно в 1.3 Пбайта в сутки! Хранить эти данные за длительный срок — непозволительно дорого.

Таким образом, для успешной реализации подходов следует обратить пристальное внимание на следующие важные моменты, лежащие в основе их реализации:

1) выбор и конфигурация системы мониторинга как источника данных;

2) осуществление, по возможности, обработки и агрегации данных "на лету";

3) использование сырых, непрореженных данных только для специальных исследований, сохраняя их отдельно от основного потока.

4. Комплексный анализ производительности суперкомпьютерных систем. Основная идея предлагаемого подхода — обеспечить возможность комплексного анализа динамики поведения суперкомпьютерной системы с разных точек зрения с учетом данных системного мониторинга при исследовании эффективности на следующих основных уровнях:

1) эффективность выполнения приложения;

2) эффективность распределения задач по очередям;

3) эффективность использования ресурсов системы.

Тем самым производится всестороннее исследование, покрывающее все основные аспекты функционирования современного суперкомпьютера в смысле эффективного выполнения пользовательских задач.

Такой комплексный подход доступен для исследователей всех уровней подготовки, относительно прост в реализации, гибок в настройке. Рассмотрим, какого рода исследования могут проводиться на каждом из трех вышеуказанных уровней.

4.1. Исследование эффективности выполнения приложения. Основная задача любой суперкомпьютерной установки — удовлетворение потребностей пользователей, решающих задачи, за которыми стоят реальные проблемы из различных прикладных областей. Критерии удовлетворения этих потребностей могут быть разными: своевременность, скорость получения результата, достижение необходимой точности расчета, обработка больших данных и т.д. [7, 8]. Простой вычислительных ресурсов, делающий их недоступными для использования, противоречит любому из таких критериев за исключением одного, о котором будет сказано позднее. Поэтому проблема повышения эффективности работы каждого отдельного приложения чрезвычайно актуальна. Ее успешное решение для отдельного приложения благоприятно сказывается на решении не только той реальной научной задачи, которая за ней стоит, но и на работе всей системы в целом.

Максимально быстрое выполнение задач. При исследовании эффективности отдельного приложения обычно [9] во главу угла ставятся скорость работы программы, минимизация простоев, максимизация загрузки процессора, полное использование ресурсов параллелизма. Это обусловлено естественным пониманием того, как можно достичь результата максимально быстро, используя все возможности программно-аппаратного комплекса. Такого рода критерии свойственны для пользователей вычислительных систем.

Минимальная стоимость выполнения задач. Однако есть и альтернативный подход, который приобретает все бóльшую популярность в свете роста масштабов вычислительных систем и соответствующего роста энергопотребления [10]. Потребление современных суперкомпьютеров измеряется мегаваттами. Стоимость потребляемой электроэнергии в таких масштабах очень велика и вполне сопоставима со сто-

имостью оборудования, составляя, при грубой оценке, около ее половины. Как известно, наиболее дорогими являются операции обмена данными и операции ввода-вывода. Таким образом, такая стратегия в общих чертах сводится к минимизации количества и суммарного объема подобного рода операций, что в большинстве случаев сводится к оптимизации расположения данных программы. Критерии минимизации энергозатрат актуальны прежде всего для владельцев систем. Для пользователей это становится актуально только при условии зависимости стоимости счета не просто от затраченных ресурсов, но и от потребленной ими энергии.

Детальное исследование конкретных свойств приложения. Еще один подход имеет характер специальных, узконаправленных исследований. Обычно это делается для поиска возможных путей оптимизации приложения и повышения эффективности его работы [11]. Суть его состоит в том, что проводится детальное рассмотрение определенной составляющей профиля выполнения программы с использованием всех доступных для этого датчиков. Такой составляющей может быть, например, работа с памятью или межузловой обмен данными. Полученные таким образом динамические характеристики приложения могут быть успешно использованы для совместного анализа с данными, полученными путем инструментирования и трассировки [12]. Учитывая возможные ограничения по числу датчиков и частоты съема данных, детальные исследования зачастую проводятся за счет сокращения числа рассматриваемых динамических характеристик, носящих общий характер и не имеющих прямого отношения к рассматриваемой составляющей профиля.

4.2. Исследование потока задач в очередях. Помимо отличий в строении и архитектуре вычислительных систем разного уровня производительности существенные отличия проявлялись и в принципах организации работы на суперкомпьютерных системах. Причиной этому была особая ценность счетного времени такой системы. Действительно, более мощная вычислительная система позволяет получить исследователю решение реальной задачи из его профессиональной области быстрее и точнее, а зачастую нужное решение просто не может быть получено на системах меньшего масштаба. Естественным образом возникали очереди на выполнение расчетов на таких системах, разрабатывались системы, позволяющие более эффективно размещать задачи разного масштаба и приоритета в очереди на счет.

Значительная часть современных суперкомпьютерных систем является неоднородной. Это возникает и вследствие ввода в эксплуатацию системы поэтапно, и вследствие обновления систем. Кроме того, зачастую системы изначально строятся неоднородными для удовлетворения потребностей разного рода приложений и сохранения разумной общей стоимости системы в целом. Конечно, разумным является подход разделения всего счетного поля на отдельные очереди по принципу однородности вычислительных ресурсов. Такой подход позволяет распределять приложения в соответствии с их спецификой. Это может быть ориентированность на использование ускорителей, в том числе графических, необходимость использования локальных дисков или большого объема оперативной памяти, наличие высокоскоростной коммуникационной сети и т.п.

Такие распределения делаются чаще всего на основании некоторой предварительной оценки. Пользователь ставит свою задачу в определенную очередь, исходя из своих соображений и того, что ему доступно и разрешено. Нередко встречаются случаи, когда даже сами авторы программ не имеют достаточного понимания специфики их поведения и особенностей, которые приложение предъявляет к программному окружению и аппаратной части вычислительной системы. В результате возможными становятся два основных типа некорректной постановки задачи в очередь:

1) постановка задачи на счет не в ту очередь из-за неправильного или неполного понимания специфики решаемой задачи, тем более что на практике свойства задачи реально отличаются от того, что от задач ожидали даже их авторы;

2) умышленная постановка задачи в очередь, которая не предназначена для данного типа задач, но является по каким-то причинам предпочтительной для пользователя (больше памяти на узлах, меньшее количество задач в очереди и т.п.).

Путем исследования средних значений динамических характеристик по задачам каждой очереди [13] достаточно хорошо прослеживается характер запускаемых задач и адекватность их разбиения по очередям. Для выбивающихся из массы случаев можно провести детальный анализ, в том числе на основе вышеописанного подхода к исследованию отдельных приложений. Такой анализ позволяет помочь в правильном распределении по специально выделенным очередям и пресечь нарушения правил распределения задач.

Взаимное влияние приложений. Интересным направлением исследований является анализ взаимного влияния одновременной работы приложений [14]. Вследствие конкуренции за тот или иной ресурс каждое приложение может оказаться в условиях нехватки ресурса существенно раньше или же вовсе не получить

ресурс в необходимом для работы объеме. Это может быть объем памяти, пропускная способность сетевого канала, интерфейс ввода-вывода и т.п.

4.3. Эффективность использования ресурсов системы. Важным результатом анализа значений динамических характеристик приложений является получение так называемых интегральных характеристик по задачам. Интегральные характеристики включают в себя средние, максимальные и минимальные значения ключевых динамических характеристик. В сочетании с данными от системы управления потоком задач о времени счета и числе задействованных узлов создается основа для определения ориентированности пользовательских приложений. При исследовании данных за достаточно большой срок (порядка месяца и более) прослеживается, насколько полно используются те или иные ресурсы системы. Например, как часто задачи упирались в пропускную способность коммуникационной сети, как часто случались простои из-за перегруженности сетевой файловой системы, какова доля задач, использующих лишь часть доступных вычислительных ядер на узле, и т.п. Такой анализ дает возможность принимать правильные решения в некоторых стратегически важных вопросах.

Уточнение политики доступа к вычислительным ресурсам. Действительно, увидев значительную долю задач, очень ограниченно использующих ресурсы параллелизма, можно принять решение о вынесении их в отдельную очередь или принять какую-то другую политику, предоставляя привилегированные условия для запуска приложений, способных в большей мере использовать ресурс суперкомпьютерной системы. Это — одно из проявлений общей большой проблемы сбалансированности использования ресурсов [15].

Принятие решений об обновлении системы. По тому, в какие ограничения упираются приложения во время работы, можно судить, какой из программных или аппаратных компонентов вычислительной системы является наиболее узким местом и нуждается в обновлении. Те же исследования актуальны и при проектировании новых систем. Интересно, что по результатам исследований можно принимать решение не только по расширению возможностей системы, но и по их сокращению вместе с сокращением ее стоимости. Вряд ли кого-то удивит ситуация, когда дорогая и быстрая коммуникационная сеть используется на мизерную долю своих возможностей. В таком случае было бы целесообразнее установить более дешевую сеть, а вырученные средства пустить на что-то более востребованное — объем памяти на узлах и т.п.

5. Набор ключевых динамических характеристик. Как уже упоминалось, не всегда имеется возможность сбора, хранения и последующей обработки значений всех доступных датчиков. Отсюда вытекает необходимость выделения минимального набора датчиков и частоты съема данных, позволяющих получить качественную картину профиля [16].

Данным набором может быть, например, следующий: загрузка процессора; число операций с плавающей точкой; число процессов, готовых принять управление (Load Average); интенсивность межузлового обмена; интенсивность ввода/вывода; число промахов при доступе к кэш-памяти. Конечно, такой список может как расширяться с добавлением к рассмотрению новые датчиков, так и быть улучшен с точки зрения частоты получения исследуемых характеристик.

Для стратегии детального исследования конкретных свойств приложения следует использовать все доступные датчики, относящиеся к соответствующей части профиля выполнения программы. К примеру, при исследовании профиля взаимодействия с памятью целесообразно рассматривать число промахов по всем уровням кэш-памяти, общее число обращений к оперативной памяти и т.п.

Для стратегии минимизации энергетической стоимости целесообразно исследовать данные о потреблении энергии вычислительными узлами и их компонентами. К сожалению, на данный момент очень немногие производители аппаратуры предоставляют доступ к такой информации. Однако необходимость такого рода исследований и критериев оптимизации следует обязательно учитывать при проектировании новых систем.

Конкретный набор ключевых характеристик может быть своим не только при разных стратегиях исследования, но и при разных реализациях систем мониторинга, обработки, агрегации и хранения данных. Действительно, и система мониторинга, и аппаратура — все вносит индивидуальные ограничения.

Важно отметить, что выбранный набор ключевых характеристик, изначально выделенный в ходе решения задачи по исследованию динамических свойств отдельной задачи, может быть основой для построения интегральных характеристик, используемых в подходах по исследованию структуры потока задач в очереди или исследованию эффективности использования вычислительных ресурсов в целом. Состав наборов может отличаться, отвечая конкретным интересам в исследованиях, однако для упрощения понимания процессов и лучшей согласованности исследуемых данных целесообразно опираться на единый набор динамических характеристик и построенные на его основе интегральные характеристики.

Идея привлечения данных системного мониторинга для углубленного анализа производительности

приложения с помощью специализированного инструментария, базирующегося на принципах инструментирования и трассировки, нашла свое воплощение, в частности, в проекте HOPSA [17]. Созданная в рамках проекта интегрированная среда позволила получить доступ к данным системного мониторинга через специальный интерфейс или через файл трассы специализированным инструментам, таким как Scalasca, Vampir и Paraver. Напомним, что применение такого рода инструментария к потоку задач невозможно ввиду ориентированности на детальное исследование отдельного приложения и необходимости внесения в него изменений либо выполнения приложения в условиях определенного состояния окружения.

Собранные данные системного мониторинга можно исследовать в автоматизированном режиме, выявляя выход за некоторые граничные значения и т.п. [18, 19]. Наглядным и удобным является представление всех данных в едином временно́м масштабе в виде графиков [20]. В таком случае хорошо прослеживаются ключевые моменты в ходе выполнения программы по одновременным, связанным изменениям.

Помимо выделения набора ключевых характеристик, следует обратить внимание на два важных аспекта, связанных с описанием и интерпретацией динамических характеристик. В значительной мере они относятся к используемой системе мониторинга, однако корректный съем данных — это исключительно важный момент, определяющий саму возможность опираться на данные системного мониторинга.

5.1. Формат описания динамических характеристик. Для корректного восстановления профиля по последовательности значений динамических характеристик возможным решением может быть составление описания каждого из соответствующих датчиков. Причин возникновения проблем с интерпретацией значений датчиков масса. Прежде всего, различные системы мониторинга имеют различные форматы описания таких датчиков. Кроме того, различные системы мониторинга изначально ориентированы на разные задачи, а потому какие-то свойства одних и тех же датчиков могут быть опущены за ненадобностью в некоторых системах мониторинга. Встречались и случаи, когда после обновления агентов системы мониторинга менялись единицы измерений, что не сразу отражалось в документации.

В качестве одного из возможных решений проблемы предлагается использовать универсальное описание датчиков системы мониторинга в формате JSON следующего вида: “name” — имя датчика; “Id” — идентификатор; “description” — понятное человеку описание назначения датчика; “unit” — единицы измерения “bytes/sec”, “%” и т.д.; “val_type” — тип значения, абсолютный или накопительный — absolute or accumulated; “scope” — когда измерено значение now, sinceLast or untilNext; “supported” — поддерживается ли в данной реализации системы мониторинга; “avail” — доступность в данный момент (может быть отключена временно); “source” — источник clustrxwatch, ganglia и др.; “granularity” — частота съема в Гц; “min” — разумный минимум области допустимых значений; “max” — разумный максимум области допустимых значений; “nodes” — список узлов, поддерживающих данную метрику.

Приведем пример для датчика, отражающего степень пользовательской загрузки ядра:

```
{
 “name”:“cpu_user”,
 “id”:1050,
 “desc”:“User CPU time”,
 “unit”:“%”,
 “val_type”:“abs”,
 “scope”:“now”,
 “supported”:1,
 “avail”:1,
 “source”:“clustrxwatch”,
 “granularity”:1,
 “min”:0,
 “max”:100,
 “nodes”:[
          “cn04”,
          “cn06”,
          “cn07”,
          “cn08”
          ]
}
```

Использование такого рода соглашений в описании динамических характеристик создает хорошие условия для интеграции различных инструментов между собой, что, безусловно, расширяет возможности при анализе профиля приложения.

5.2. Диапазон возможных значений характеристик. Помимо корректного описания динамических характеристик следует обратить внимание еще на один момент. С одной стороны, важно правильно интерпретировать полученные значения. С другой — важно иметь возможность вовремя отследить ситуацию, при которой данные, продолжая поступать, перестают соответствовать реальности. Такое возможно, например, после обновления ПО, относящегося к системе мониторинга, а не только в связи со сбоем. Существенную помощь в таких вопросах может оказать заранее составленное описание диапазонов допустимых значений для каждой ключевой характеристики. Ситуации, при которых значения характеристик выходят за рамки диапазона допустимых значений, будем называть артефактами. Часто такие ситуации вызваны переполнениями счетчиков. Подобного вида эффекты следует исключать из расчетов интегральных значений. Тем не менее, в некоторых случаях после исследования причин возникновения такое значение может быть приведено к нормальному. Приведем пример для вышеуказанного набора ключевых характеристик и суперкомпьютерной системы "Чебышев" Московского университета.

Загрузка процессора "CPU_user". Мера измерения — доля (или в процентах) загрузки процессора пользовательской задачей. Усреднение — по ядрам (табл. 1).

Таблица 1

Базовые признаки поведения на основании значений CPU_user

Значение	Характерная трактовка	Пример
0	Процессор не загружен пользовательской задачей	0
1	Полностью занят пользовательской задачей	1
Более 0.75	Высокая загрузка процессора	80%
Менее 0.1	Низкая загрузка процессора	5%
Менее 0	Артефакт	-0.5
Более 1	Артефакт	120%

Число операций с плавающей точкой (flop/s). Число операций с плавающей точкой. В зависимости от реализации система сбора данных системного мониторинга и сложные инструкции (например, SSE) могут учитываться как одна операция и т.п. Усреднение — по ядрам (табл. 2).

Таблица 2

Базовые признаки поведения на основании значений flop/s

Значение	Характерная трактовка	Пример
0	Отсутствует арифметика с плавающей точкой. Характерно для задач, ориентированных на целочисленные вычисления (задачи поиска и т.д.)	0
Peak = Частота_процессора × число_операций_за_такт	Максимальное значение, полная загрузка	12 Gflop/s ("Чебышев")
Менее 0	Артефакт	-10^6
Более Peak	Артефакт	10^{15}
Более 0.75 × Peak	Высокая интенсивность вещественной арифметики	10 Gflop/s ("Чебышев")
Менее 0.1 × Peak	Низкая интенсивность вещественной арифметики	1 Gflop/s ("'Чебышев")

Число готовых к работе процессов (Load Average). Число готовых к взятию управления процессов, включая работающие. Усреднение — по узлам (табл. 3).

Скорость передачи данных по транспортной сети. Объем переданных или принятых данных за единицу времени (Мбит/с). Усреднение — по узлам (табл. 4).

Скорость передачи данных по коммуникационной сети. Объем переданных или принятых данных за единицу времени (Мбит/с). Усреднение — по узлам (табл. 5).

Число промахов при доступе к кэш первого уровня (L1 Cache misses). Число промахов в кэш данных первого уровня (промахов/с). Усреднение — по ядрам (табл. 6).

Таблица 3

Базовые признаки поведения на основании значений Load Average

Значение	Характерная трактовка	Пример
0	Нет готовых к взятию управления процессов	0
$N \times K$ для узла из K штук N-ядерных процессоров	Полная загрузка процессоров при отсутствии конкуренции процессов. Идеальный вариант	8 (“Чебышев”)
На 1–2 ниже N	Высокая, но не полная загрузка	6 (“Чебышев”)
$N < x < 1.5N$ для N-ядерного процессора	Повышенная конкуренция между процессами	9 (“Чебышев”)
Много более N	Очень большая конкуренция, процессов больше, чем ресурсов, часть всегда будет простаивать. Кроме того, частая смена контекстов	25 (“Чебышев”)
Менее 0	Артефакт	-3

Таблица 4

Базовые признаки поведения на основании значений датчика интенсивности обмена по сети Ethernet

Значение	Характерная трактовка	“Чебышев”
0	Сеть простаивает	0
tr_max (максимальная физическая пропускная способность сети)	Полная загрузка сети	1 Гбит/с (ETH_send и ETH_receive “Чебышев”)
Менее 0.2 tr_max	Низкая интенсивность передачи	100 Мбит/с (ETH_send и ETH_receive “Чебышев”)
Более 0.5 tr_max	Высокая интенсивность передачи	750 Мбит/с (ETH_send и ETH_receive “Чебышев”), де-факто ограничен производительностью сетевого файлового хранилища
Менее 0	Артефакт	-10^6
Более tr_max	Артефакт	10^{10} (ETH_send и ETH_receive “Чебышев”)
Преобладание ETH_send / ETH_receive	Вычислительные свойств задач: отношение объема входных и выходных данных (иногда и вре́менных данных)	

6. База причин и признаков. Необходимо особо подчеркнуть важность работ, направленных на углубление понимания причин снижения производительности и на выявление их признаков в виде характерного поведения динамических характеристик. Например, в Московском университете ведутся работы по созданию банка примеров неэффективного поведения суперкомпьютерных приложений. Создание таких банков знаний, безусловно, повысит качество анализа специалистами и сделает подход к анализу в целом более доступным для не обладающих большим опытом пользователей. При этом выбор дальнейшего направления анализа и оптимизации приложения [21] будет с большей вероятностью правильным, так как будет более понятна природа причин снижения эффективности в приложении пользователя.

Для решения указанной задачи была разработана модель представления таких примеров и создан прототип базы данных. В основе реализации лежит проверенная связка веб-сервера Apache и СУБД MySQL.

База данных состоит из нескольких таблиц:

‘apps’ — содержащит общую информацию о приложении;

‘cases’ — содержащит соответствие найденного отдельного признака истинным причинам;

‘metrics’ — список динамических характеристик;

Таблица 5

Базовые признаки поведения на основании значений датчика интенсивности обмена по сети Infiniband

Значение	Характерная трактовка	“Чебышев”
0	Сеть простаивает	0
com_max (максимальная физическая пропускная способность сети)	Полная загрузка сети	16 Гбит/с (IB_send и IB_receive “Чебышев”)
Менее 0.2 com_max	Низкая интенсивность передачи	100 Мбит/с (IB_send и IB_receive “Чебышев”)
Более 0.5 com_max	Высокая интенсивность передачи	9 Гбит/с (IB_send и IB_receive “Чебышев”)
Менее 0	Артефакт	-10^6
Более com_max	Артефакт	10^{11} (IB_send и IB_receive “Чебышев”)

Таблица 6

Базовые признаки поведения на основании значений счетчика промахов при обращении в кэш первого уровня

Значение	Характерная трактовка	“Чебышев”
0	Не было обращений к памяти вообще или все обращения свелись к обращениям к кэш. Реально не встречается	0
L1_max Частота процессора / число тактов на обращение в L2	Любое обращение в память вызывает промах в L1	$\approx 200 \times 10^6$ (“Чебышев”: 3×10^9 тактов / ≈ 15 тактов)
Менее 2×10^6 (на примере системы “Чебышев”)	Низкая интенсивность промахов	10^6 (“Чебышев”)
Более 20×10^6 (на примере системы “Чебышев”)	Высокая интенсивность промахов	50×10^6 (“Чебышев”)
Менее 0	Артефакт	-10^3
Более L1_max	Артефакт	10^{10} (“Чебышев”)

‘root_cpu’ — формулировка причин, относящихся к использованию CPU;
‘root _io’ — формулировка причин, относящихся к использованию ввода/вывода;
‘root _mem’ — формулировка причин, относящихся к использованию памяти;
‘root _net’ — формулировка причин, относящихся к использованию сетевого обмена;
‘symptoms’ — формулировки признаков в терминах динамических характеристик.

Для каждого исследуемого запуска приложения заводится запись в таблице “apps” в виде
‘id’ — уникальный идентификатор приложения;
‘app_name’ — имя приложения;
‘app_desc’ — описание приложения;
‘code’ — ссылка на код программы;
‘run_t1’ — время начала работы программы;
‘run_t2’ — время окончания работы программы;
‘run_system’ — имя системы, на которой велся счет;
‘run_nnodes’ — число выделенных узлов;

‘run_ncores’ — число задействованных вычислительных ядер;
‘run_line’ — строка запуска;
‘issues’ — описание сути проблем или особенностей приложения;
‘checked’ — индикатор проверенной записи.

Каждое необычное событие в ходе выполнения приложения описывается отдельной записью в таблице “cases”:

‘id’ — уникальный идентификатор записи;
‘case_id’ — идентификатор рассматриваемого случая;
‘time1’ — время начала временного фрейма;
‘time2’ — время завершения рассматриваемого временного фрейма;
‘image’ — ссылка на график или диаграмму;
‘metric’ — ссылка на конкретную ключевую динамическую характеристику;
‘symptom’ — ссылка на формулировку поведения характеристики;
‘cause_cpu’ — формулировка причины из области использования CPU;
‘cause_mem’ — формулировка причины из области использования памяти;
‘cause_net’ — формулировка причины из области использования сети;
‘cause_io’ — формулировка причины из области использования ввода/вывода;
‘comment’ — комментарии;
‘checked’ — индикатор проверенной записи.

Под поведением характеристики понимаем следующие характерные ситуации:

“полка” — неизменный (с погрешностью) уровень последних значений;
“рост” — увеличение значений (быстрый рост и т.п.);
“падение” — уменьшение значений (быстрое падение и т.п.);
“максимум” — преодоление некоторого максимально возможного порога;
“минимум” — преодоление некоторого минимально возможного порога;
“всплеск” — временное существенное отклонение от “полки”;
“скачок” — изменение “полки”;
“шум” — невыявленная закономерность изменений значений;
“осцилляция” — частое изменение, скачки значений на коротком интервале времени.

Таким образом, для каждого случая указывается временной интервал, динамическая характеристика, характер ее поведения, формулировка выявленного признака и ссылка на соответствующую причину такого поведения. Дополнительно информация сопровождается графиками и общим описанием приложения. В ходе дальнейших работ, возможно, структура будет расширена. Основываясь на опыте исследований профилей приложений по данным системного мониторинга, можно утверждать, что даже в таком базовом виде создаваемый банк примеров будет исключительно полезен широкому кругу специалистов — от начинающих до экспертов.

В качестве интерфейса на данный момент используется общедоступное средство работы phpMyAdmin. Создание пользовательского интерфейса, предоставляющего доступ к примерам неэффективного поведения программ и их детальному описанию (значения характеристик, графики, текстовые описания), будет завершено к середине 2014 г. К концу года поставлена задача наполнить созданную базу примерами. Примеры планируется использовать как синтетические, заведомо неэффективные, так и примеры разбора реальных пользовательских приложений и встреченных в них случаев необычного поведения.

7. Возникающие вопросы и проблемы. Исследование эффективности на основе системного мониторинга на хорошем уровне предъявляет серьезные требования к технологиям, на которые оно опирается, а также ставит некоторые новые вопросы, ответы на которые надо получить.

Прежде всего, очень важной является основа — сама система мониторинга, которая будет предоставлять данные для анализа. Это отдельная большая тема для исследований. Однако такие основополагающие вопросы, как масштабируемость во всех смыслах, гибкость настройки, широта спектра собираемых характеристик, — необходимо упомянуть.

Неотрывно от системы мониторинга стоит вопрос организации обработки данных “на лету”, агрегации и последующего сохранения для апостериорного анализа. Вместе с тем, сама организация сохранения ранее собранных данных и доступа к ним, в том числе в виде подготовленных отчетов, прошедших некоторый предварительный анализ, также требует внимательного рассмотрения.

Несколько в стороне стоит проблема доступности некоторых важных характеристик. Прежде всего это касается данных об энергопотреблении на уровне узла в целом или отдельными ядрами или интерфейсами. Хочется надеяться, что общая озабоченность энергетически эффективными решениями выльется

в более широкое предоставление производителями данных о потреблении аппаратуры в виде доступных для считывания датчиков.

Конечно же, этим не ограничивается перечень сопряженных с основной темой вопросов, их очень много, все они тесно связаны между собой, и каждый представляет достаточный интерес и важность для отдельного рассмотрения.

8. Заключение. В настоящей статье обосновывается необходимость комплексного подхода к исследованию эффективности функционирования суперкомпьютерной системы. Выделены три основных уровня абстракции:

— оценка интегрального использования ресурсов вычислительной системы;

— исследование распределения по потокам задач;

— анализ эффективности и особенности работы отдельного приложения.

Реализация каждого уровня может отличаться для разных систем, разных масштабов и прочих условий. Важнейшими сопряженными вопросами при реализации являются

— система мониторинга как источник данных;

— определение набора ключевых характеристик и требуемой гранулярности;

— обработка данных “на лету”;

— выявление и описание признаков причин потенциального снижения эффективности;

— система хранения и доступа к ранее собранным данным;

— накопление экспериментальных данных, например в виде создания банка примеров неэффективного поведения программ.

Продемонстрирован вариант выделения набора ключевых динамических характеристик, предложен формат описания соответствующих им датчиков, приведены примеры описания диапазонов допустимых значений.

Показана возможная организации банка примеров неэффективного поведения суперкомпьютерных приложений на примере разрабатываемого инструментария в Суперкомпьютерном комплексе МГУ им. М. В. Ломоносова.

Применение этого комплексного подхода позволит существенно повысить эффективность работы суперкомпьютерных систем. Апробация предложенного подхода ведется на Суперкомпьютерном комплексе МГУ.

Работа выполняется при финансовой поддержке РФФИ (проекты 13–07–00786 и 13–07–12206–офи_м).

список литературы

1. *Воеводин Вл.В., Жуматий С.А.* Вычислительное дело и кластерные системы. М.: Изд-во Моск. ун-та, 2007.
2. *Hennessy J., Patterson D.* Computer architecture. San Francisco: Morgan Kaufmann, 2011.
3. *Grama A., Gupta A., Karypis G., Kumar V.* Introduction to parallel computing. Reading: Addison-Wesley, 2003.
4. *Hennessy J., Patterson D.* Computer organization and design. The hardware/software interface. San Francisco: Morgan Kaufmann, 2008.
5. *Никитенко Д.А., Стефанов К.С.* Исследование эффективности параллельных программ по данным мониторинга // Вычислительные методы и программирование. 2012. **13**. 97–102.
6. *Воеводин Вл.В., Жуматий С.А., Соболев С.И., Антонов А.С., Брызгалов П.А., Никитенко Д.А., Стефанов К.С., Воеводин Вад.В.* Практика суперкомпьютера “Ломоносов” // Открытые системы. 2012. № 7. 36–39.
7. *Bekakos M. (Ed.)* Highly parallel computations: algorithms and applications. Southampton: WIT Press, 2001.
8. *Berry M., Gallivan K., Gallopoulos E., Grama A., Philippe B., Saad Y., Saied F. (Eds.)* High-performance scientific computing. Algorithms and applications. New York: Springer, 2012.
9. *Адинец А.В., Брызгалов П.А., Воеводин Вад.В., Жуматий С.А., Никитенко Д.А.* Об одном подходе к мониторингу, анализу и визуализации потока заданий на кластерной системе // Вычислительные методы и программирование. 2011. **12**. 90–93.
10. *Jin X., Zhang F., Song Y., Fan L., Liu Z.* Energy: efficient scheduling with time and processors eligibility restrictions // Lecture Notes in Computer Science. Vol. 8097. Heidelberg: Springer, 2013. 66–77.
11. *Bailey D., Lucas R., Williams S. (Eds.)* Performance tuning of scientific applications. Boca Raton: CRC Press, 2011.
12. *Servat H., Llort G., Gimenez J., Huck K., Labarta J.* Folding: detailed analysis with coarse sampling // Tools for High Performance Computing. Heidelberg: Springer, 2013. 105–118.
13. *Антонов А.С., Жуматий С.А., Никитенко Д.А., Стефанов К.С., Теплов А.М., Швец П.А.* Исследование динамических характеристик потока задач суперкомпьютерной системы // Вычислительные методы и программирование. 2013. **14**. 104–108.
14. *Shah A., Wolf F., Zhumatiy S., Voevodin Vl.* Capturing inter-application interference on clusters // Proc. of the 2013 IEEE Int. Conf. on Cluster Computing (CLUSTER 2013). New York: IEEE Press, 2013. 1–5.

15. *Bohme D., Geimer M., Wolf F.* Characterizing load and communication imbalance in large-scale parallel applications // Proc. of the 26th IEEE Int. Parallel & Distributed Processing Symposium (IPDPS). New York: IEEE Press, 2013. 2538–2541.
16. *Treibig J., Hager G., Wellein G.* Best practices for HPM-assisted performance engineering on modern multicore processors // Lecture Notes in Computer Science. Vol. 7640. Heidelberg: Springer, 2013. 451–460.
17. *Mohr B., Voevodin Vl., Gimenez J., Hagersten E., Knuepfer A., Nikitenko D., Nilsson M., Servat H., Shah A., Winkler F., Wolf F., Zhujov I.* The HOPSA workflow and tools // Tools for High Performance Computing. Heidelberg: Springer, 2013. 127–146.
18. *Андреев Д.Ю., Антонов А.С., Воеводин Вад. В., Жуматий С.А., Никитенко Д.А., Стефанов К.С., Швец П.А.* Система автоматизированного поиска ошибок и неэффективностей в параллельных программах // Вычислительные методы и программирование. 2013. **14**. 48–53.
19. *Антонов А.С., Воеводин Вад.В., Жуматий С.А., Никитенко Д.А., Стефанов К.С., Швец П.А.* Автоматизация поиска ошибок и неэффективностей в параллельных программах // Вычислительные методы и программирование. 2013. **14**. 11–17.
20. *Адинец А.В., Брызгалов П.А., Воеводин Вад.В., Жуматий С.А., Никитенко Д.А., Стефанов К.С.* Job Digest — подход к исследованию динамических свойств задач на суперкомпьютерных системах // Вестн. Уфимского гос. авиационного технического ун-та. 2013. **17**, № 2. 131–137.
21. *Ciegis R., Henty D., Kagstrom B., Zilinskas J. (Eds.)* Parallel scientific computing and optimization. Advances and Applications. Series: Springer Optimization and Its Applications. Vol. 27. Heidelberg: Springer, 2009.

Поступила в редакцию
10.01.2014

Overall Supercomputer Performance Analysis Based on System Monitoring Data

D. A. Nikitenko [1]

[1] *Research Computing Center, Lomonosov Moscow State University; Leninskie Gory, Moscow, 119992, Russia; Scientist, e-mail: dan@parallel.ru*

Received January 10, 2014

Abstract: Thorough investigation of supercomputer resource utilization efficiency is of great practical importance. Every HPC system holder and every user face such a problem. An approach to overall performance analysis, including peculiarities of application runs, assignment of jobs to queues, and total resource utilization of supercomputer systems is proposed. This approach is based on the analysis of system monitoring data and is aimed at providing a number of means for the qualitative behavior evaluation of supercomputer applications and HPC systems as a whole.

Keywords: supercomputer, performance, efficiency, parallel programs, dynamic characteristics, workload, system monitoring.

References

1. Vl. V. Voevodin and S. A. Zhumatii, *Computing and Cluster Systems* (Mosk. Gos. Univ., Moscow, 2007) [in Russian].
2. J. Hennessy and D. Patterson, *Computer Architecture* (Morgan Kaufmann, San Francisco, 2011).
3. A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing* (Addison-Wesley, Reading, 2003).
4. J. Hennessy and D. Patterson, *Computer Organization and Design. The Hardware/Software Interface.* (Morgan Kaufmann, San Francisco, 2008).
5. D. A. Nikitenko and K. S. Stefanov, "Analyzing the Parallel Program Efficiency Based on Monitoring Data," Vychisl. Metody Programm. **13**, 97–102 (2012).
6. Vl. V. Voevodin, S. A. Zhumatii, S. I. Sobolev, et al., "The Lomonosov Supercomputer in Practice," Otkrytye Sistemy, No. 7, 36–39 (2012).
7. M. Bekakos (Ed.), *Highly Parallel Computations: Algorithms and Applications* (WIT Press, Southampton, 2001).

8. M. Berry, K. Gallivan, E. Gallopoulos, A. Grama, B. Philippe, Y. Saad, and F. Saied (Eds.), *High-Performance Scientific Computing. Algorithms and Applications* (Springer, New York, 2012).

9. A. V. Adinets, P. A. Bryzgalov, Vad. V. Voevodin, et al., "An Approach to Cluster System Task Flow Monitoring, Analysis, and Visualization," Vychisl. Metody Programm. **12**, 90–93 (2011).

10. X. Jin, F. Zhang, Y. Song, et al., "Energy: Efficient Scheduling with Time and Processors Eligibility Restrictions," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2013), Vol. 8097, pp. 66–77.

11. D. Bailey, R. Lucas, and S. Williams (Eds.), *Performance Tuning of Scientific Applications* (CRC Press, Boca Raton, 2011).

12. H. Servat, G. Llort, J. Gimenez, et al., "Folding: Detailed Analysis with Coarse Sampling," in *Tools for High Performance Computing* (Springer, Heidelberg, 2013), pp. 105–118.

13. A. S. Antonov, S. A. Zhumatii, D. A. Nikitenko, et al., "Examination of Supercomputer System Jobs Flow Dynamic Characteristics," Vychisl. Metody Programm. **14**, 104–108 (2013).

14. A. Shah, F. Wolf, S. Zhumatiy, and Vl. Voevodin, "Capturing Inter-Application Interference on Clusters," in *Electronic Proc. 2013 IEEE Int. Conf. on Cluster Computing* (IEEE Press, New York, 2013), pp. 1–5.

15. D. Bohme, M. Geimer, and F. Wolf, "Characterizing Load and Communication Imbalance in Large-Scale Parallel Applications," in *Electronic Proc. 26th IEEE Int. Parallel & Distributed Processing Symposium* (IEEE Press, New York, 2013), pp. 2538–2541.

16. J. Treibig, G. Hager, and G. Wellein, "Best Practices for HPM-Assisted Performance Engineering on Modern Multicore Processors," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2013), Vol. 7640, pp. 451–460.

17. B. Mohr, Vl. Voevodin, J. Gimenez, et al., "The HOPSA Workflow and Tools," in *Tools for High Performance Computing* (Springer, Heidelberg, 2013), pp. 127–146.

18. D. Yu. Andreev, A. S. Antonov, Vad. V. Voevodin, et al., "A System for the Automated Finding of Inefficiencies and Errors in Parallel Programs," Vychisl. Metody Programm. **14**, 48–53 (2013).

19. A. S. Antonov, Vad. V. Voevodin, S. A. Zhumatii, et al., "Automating the Location of Errors and Inefficiencies in Parallel Programs," Vychisl. Metody Programm. **14**, 11–17 (2013).

20. A. V. Adinets, P. A. Bryzgalov, Vad. V. Voevodin, et al., "Job Digest: An Approach to Study Dynamic Properties of Tasks on Supercomputing Systems," Vestn. Ufimsk. Gos. Aviats. Tekhn. Univ. **17** (2), 131–137 (2013).

21. R. Ciegis, D. Henty, B. Kagstrom, and J. Zilinskas (Eds.), *Parallel Scientific Computing and Optimization* (Springer, Heidelberg, 2009), Adv. and Appl. Ser.: Springer Optim. and Its Appl. Vol. 27.