JOB DIGEST — ПОДХОД К ИССЛЕДОВАНИЮ ДИНАМИЧЕСКИХ СВОЙСТВ ЗАДАЧ НА СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМАХ

А.В. Адинец, П.А. Брызгалов, Вад.В. Воеводин, С.А. Жуматий, Д.А. Никитенко, К.С. Стефанов

Вместе с ростом масштаба вычислительных систем и решаемых на них задач растет и сложность написания эффективных программ. Причина этого кроется в том, что также возрастает и множество факторов, которые могут влиять на эффективность приложений. Свойства аппаратного и программного обеспечения суперкомпьютера, свойства самой исполняемой программы, взаимное влияние исполняемых программ друг на друга — все это необходимо учитывать, если стремишься добиться высокой эффективности. При этом с развитием области высокопроизводительных вычислений все эти свойства становятся только сложнее, обрастают все новыми факторами. Самые мощные современные кластеры обладают миллионами ядер; как в принципе можно организовать эффективные вычисления такого масштаба? Иерархия памяти с каждым годом становится все сложнее; как добиться ее эффективного использования?

Низкая эффективность выполняемых программ — это не единственная проблема в суперкомпьютерной области. Также важно научиться оценивать и повышать эффективность использования самого суперкомпьютера, т.е. насколько эффективно используется процессорное время и другие ресурсы вычислительного комплекса. Наш опыт показывает, что зачастую такие механизмы как алгоритмы планирования выполнения задач и квоты на использование ресурсов суперкомпьютера приводят к снижению эффективности его работы.

Все это приводит к необходимости создания инструмента, который позволит разобраться, где, а главное почему происходит потеря производительности при выполнении программ и использовании суперкомпьютеров. Понятно, что для получения качественной и многосторонней оценки этот инструмент должен обладать множеством различных данных как о поведении самой задачи, так и о состоянии суперкомпьютера в целом. Подобный инструмент разрабатывается в лаборатории Параллельных информационных технологий Научно-исследовательского вычислительного центра Московского государственного университета имени М.В. Ломоносова и получил название LAPTA — «**L**apta is a p**A**ckage for **P**erformance moni**T**oring and **A**nalysis». Данный инструмент создается в рамках выполнения совместного российско-европейского проекта HOPSA (HOlistic Performance System Analysis) [1,2].

В данной статье мы расскажем о разрабатываемом комплексе LAPTA и подробно остановимся на одном из используемых в нем подходов, который предназначен для исследования поведения задачи во время выполнения. Данный подход изучает динамические свойства задач, исследуемые с помощью средств мониторинга. Его цель состоит в предоставлении как администратору системы, так и пользователю базовых характеристик задачи по ее завершению для получения как качественной, так и детальной оценки каждого отдельно взятого запуска. Данный подход, а также полученный в результате его применения отчет получили название «Job digest».

1. Архитектура комплекса LAPTA

Комплекс LAPTA предназначен для всестороннего анализа динамических характеристик параллельных программ и суперкомпьютеров. При проведении анализа учитываются характеристики задачи, начиная от момента постановки в очередь и заканчивая ее завершением. Это позволяет получать полную информацию как о самой задаче, так и о всей совокупности выполняемых на суперкомпьютере задач.

Общая схема комплекса LAPTA представлена на рис. 1. На узлах кластера данные с различных аппаратных датчиков собираются при помощи агентов и передаются на уровень агрегации. При помощи соответствующих агентов данные собираются и с системы управления потоком задач (СУПЗ). Модули агрегации сохраняют собранные данные в базы данных при помощи модулей БД. Каждый модуль БД поддерживает работу с одним типом БД. Далее, данные из БД могут быть извлечены для анализа при помощи модулей анализа.

Задача модулей анализа — это выполнение обработки данных. При этом выполнение анализа может быть инициализировано из самых разнообразных частей системы. Например, запрос на анализ может прийти из модуля визуализации через центр обработки запросов (ЦОЗ) в результате работы пользователя с системой через веб-браузер. В этом случае основной целью запроса может быть анализ динамики поведения параллельной программы. Анализ данных может быть инициирован системными процессами, например, по таймеру раз в сутки для построения ежедневного отчёта о работе суперкомпьютера. Или же данные могут быть запрошены внешними системами интеграции и визуализации. В любом случае, инициатор запроса сначала обращается к серверу управления, который при необходимости порождает модули анализа и отслеживает их работу. Через сервер управления данные не идут — их возвращает модуль анализа напрямую инициатору запроса.

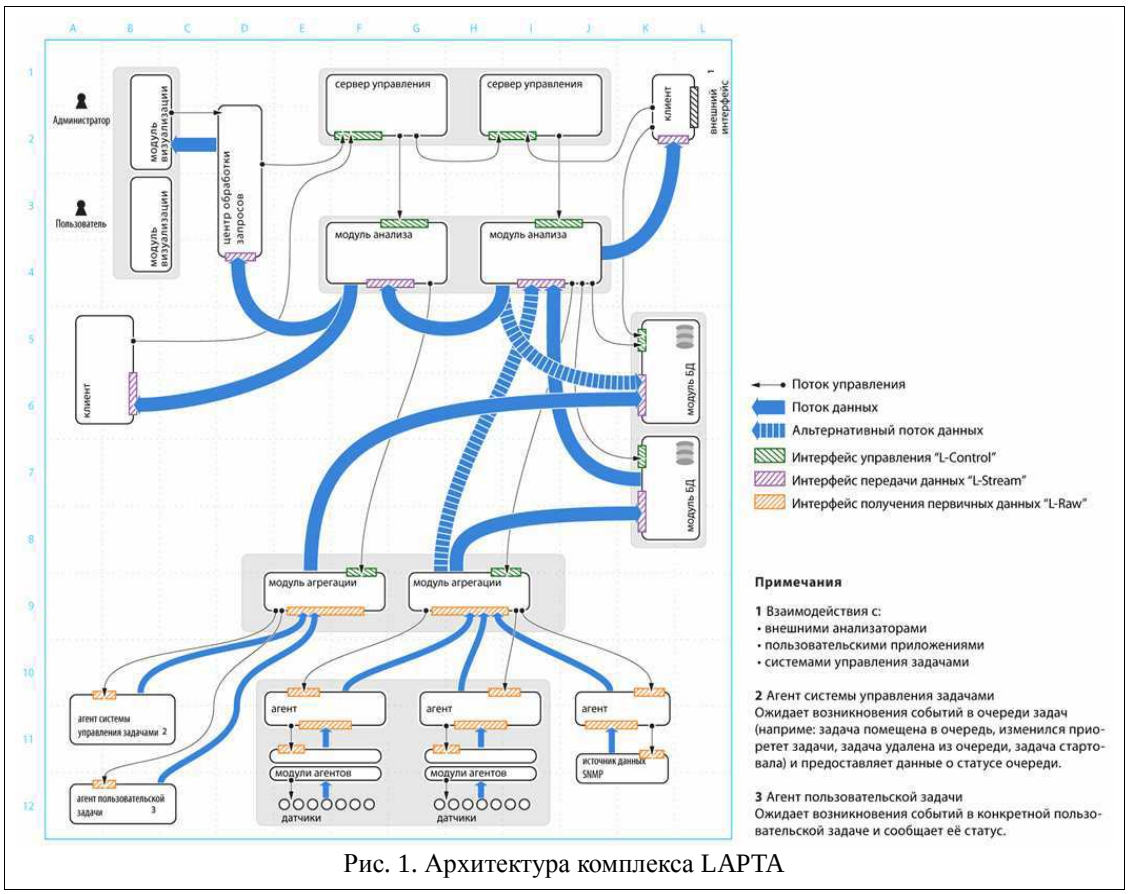

Рис. 1. Архитектура комплекса LAPTA

2. Организация подхода Job digest

В данном разделе мы опишем, каким образом в рамках комплекса LAPTA организована работа модуля, реализующего подход Job digest.

В момент постановки задачи на счет происходит либо запуск агента, который будет отслеживать изменение статуса задачи в потоке служебной информации от системы управления потоком задач, либо передается идентификатор задачи, если такой агент уже запущен. По окончании работы задачи агент выделяет в потоке вывода СУПЗ признак ее завершения и предоставляет базовую информацию (списки узлов, временной диапазон и т.п.) для дальнейшего формирования отчета Job digest. Подробнее о самом формировании отчета будет рассказано далее отдельно.

Система управления потоком задач ставит задачу на счет и передает на вывод изменения в статусе задачи. В данный момент поддерживаются СУПЗ CLEO и Slurm. Одновременно, в постоянном режиме работает система сбора системных данных, данные сохраняются, возможно, с некоторой степенью агрегации для дальнейшего апостериорного анализа. Помимо основных данных самой системы мониторинга могут собираться данные и от других источников, например, от существующих средств сбора трасс приложений. Для этого предусмотрены соответствующие интерфейсы в системе сбора и сохранения данных статистики.

Данный подход предполагает наличие инфраструктуры для доступа к сохраненным данным системного мониторинга. В частности, такая инфраструктура разрабатывается российской стороной в рамках упомянутого выше совместного проекта HOPSA. Например, разрабатываемый в рамках проекта язык Hoplang [3,4] предназначен для формирования запросов данных мониторинга.

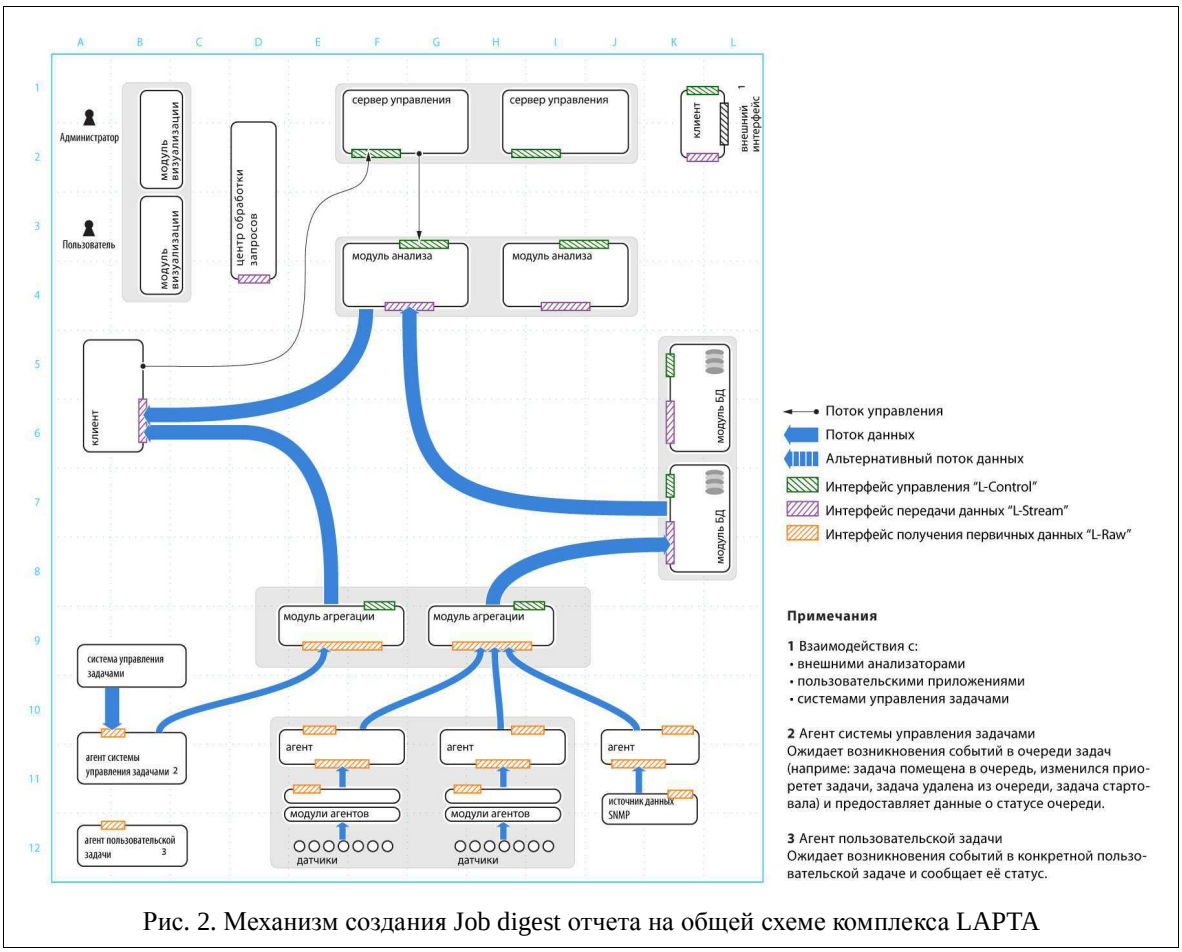

Рис. 2. Механизм создания Job digest отчета на общей схеме комплекса LAPTA

На рис.2 схематично показан процесс формирования отчета на общей схеме комплекса сбора, хранения и обработки данных мониторинга. В качестве СУБД для хранения данных мониторинга используются базы данных Cassandra и MongoDB. Модуль доступа к БД реализован в двух вариантах: на технологиях Pig и Hadoop и на упомянутой выше технологии Hoplang. И модуль, использующий технологию Pig+Hadoop, и модуль, использующий Hoplang, написаны на языке программирования Ruby. Для описания запросов на получение данных в первом случае используется язык Pig latin, а во втором – Hoplang.

3. Формирование отчета Job digest

В текущей реализации отчет Job digest не генерируется автоматически; он создается только в случае необходимости по запросу пользователя. Для составления отчета по определенному событию вызывается скрипт, который формирует запрос к Java сервлету. По умолчанию запуск скрипта производится в ручном режиме, но он может производиться, например, по окончании работы задачи.

В запросе указываются характеристики завершённой задачи, а также шаблон для формирования отчёта. Если Job digest уже был ранее сформирован (ответ от Java сервлета положителен), то информация об отчёте записывается в HTML-файл, который потом доступен через Web-браузер. Адрес этого отчёта охраняется в отдельном файле вместе с отчётом о задаче, и пользователь может впоследствии открыть его в браузере.

Java сервлет выполняет набор запросов с параметрами, и полученные данные записываются в CSV-файлы. Набор запросов и параметры к ним указываются в шаблоне, который представляет собой HTML-файл со ссылками на запросы. Сами запросы хранятся в текстовых файлах, называемых «шаблоны запросов». Запросы выполняются по очереди в том порядке, в каком они расположены в HTML-шаблоне. По окончании обработки запросов генерируется отчёт. Он получается из HTML-шаблона путём замены ссылок на шаблоны запросов специальными ссылками на CSV-файлы с результатами. Эти ссылки строятся таким образом, что при открытии отчёта в браузере на месте ссылок отображаются диаграммы и таблицы с данными, полученными из соответствующих CSV-файлов.

Общая последовательность обращений внутри модуля визуализации приведена на рис. 3.

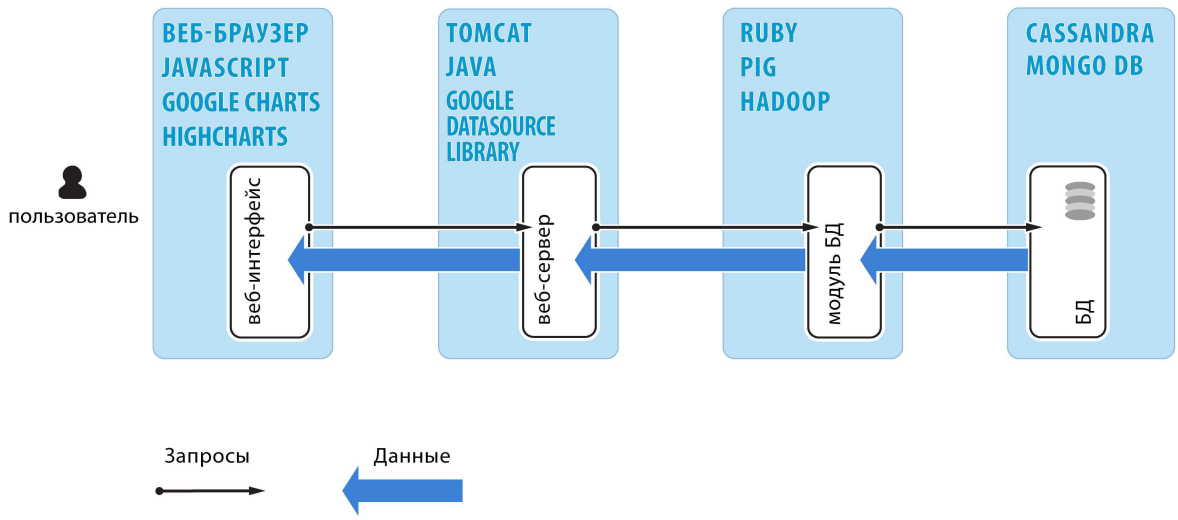

Рис. 3. Последовательность обращений при построении Job digest, используемые технологии

Web-сервер в терминах схемы на рис. 4 реализован на языке программирования Java в виде набора сервлетов, работающих под управлением web-сервера Tomcat. Web-сервер общается с модулем БД по протоколу AVRO по технологии удалённого вызова процедур (RPC). Модуль БД принимает запросы от web-сервера и возвращает данные в формате CSV.

Полученные от модуля БД данные web-сервер дополнительно обрабатывает. Для этого используется библиотека Google Datasource Library, которая позволяет использовать язык запросов Google query language для постобработки полученных от модуля БД данных. Также эта библиотека используется для форматирования данных перед их записью в CSV файлы. Визуализация данных происходит в web-браузере. Для визуализации используется JavaScript и библиотеки jQuery и Highcharts.

Динамические характеристики запусков приложений пользователей доступны администраторам системы в полном объеме, а обычным пользователям доступ предоставляется только к запускам собственных приложений. Пользователям предлагается страница (рис. 4) со списком их приложений, завершивших работу на вычислительной системе. Возле каждого приложения в списке находится ссылка для формирования отчёта по работе данного приложения, если он не был сформирован ранее, и ссылка на сам отчет, если он уже был однажды посчитан.

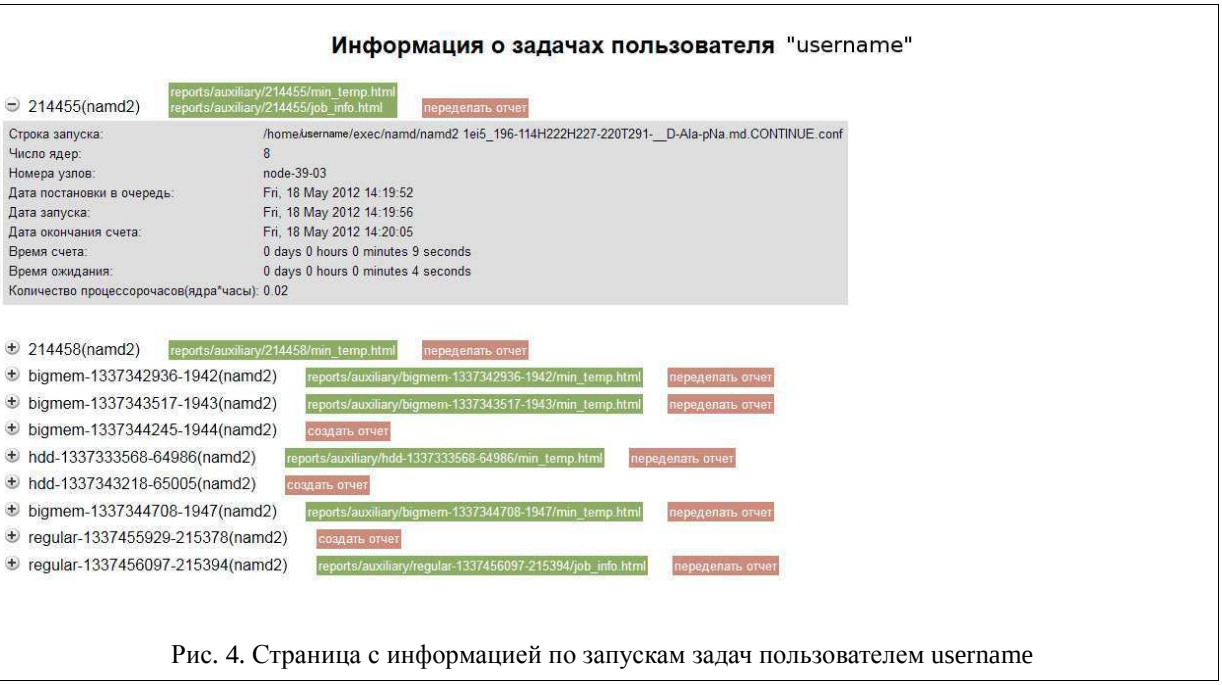

Рис. 4. Страница с информацией по запускам задач пользователем username

По окончании формирования отчёта, пользователь может просмотреть его в браузере. Основу отчета составляет информация о данных мониторинга выбранного приложения. В отчёт можно включить любое количество графиков и диаграмм, отражающих различные параметры работы приложения — загрузка процессора (рис. 5), использование памяти (рис. 6), свопинг, Load Average (рис. 6), дисковые операции, загруженность сети Ethernet (рис. 5), InfiniBand и т.д..

Данные графики уже позволяют делать содержательные выводы о поведении программы. Например, на рис. 5 видно, что загрузка CPU в начале выполнения программы практически нулевая. При этом сеть Ethernet не загружена (аналогичная картина и с сетью Infiniband), а это, скорее всего, означает, что в начале программы происходит чтение входных данных. Из этого пользователь может сделать вывод, что начальная загрузка данных занимает достаточно много времени, и может попытаться оптимизировать соответствующий фрагмент программы.

Рис. 5. Профиль загрузки CPU и сети Ethernet

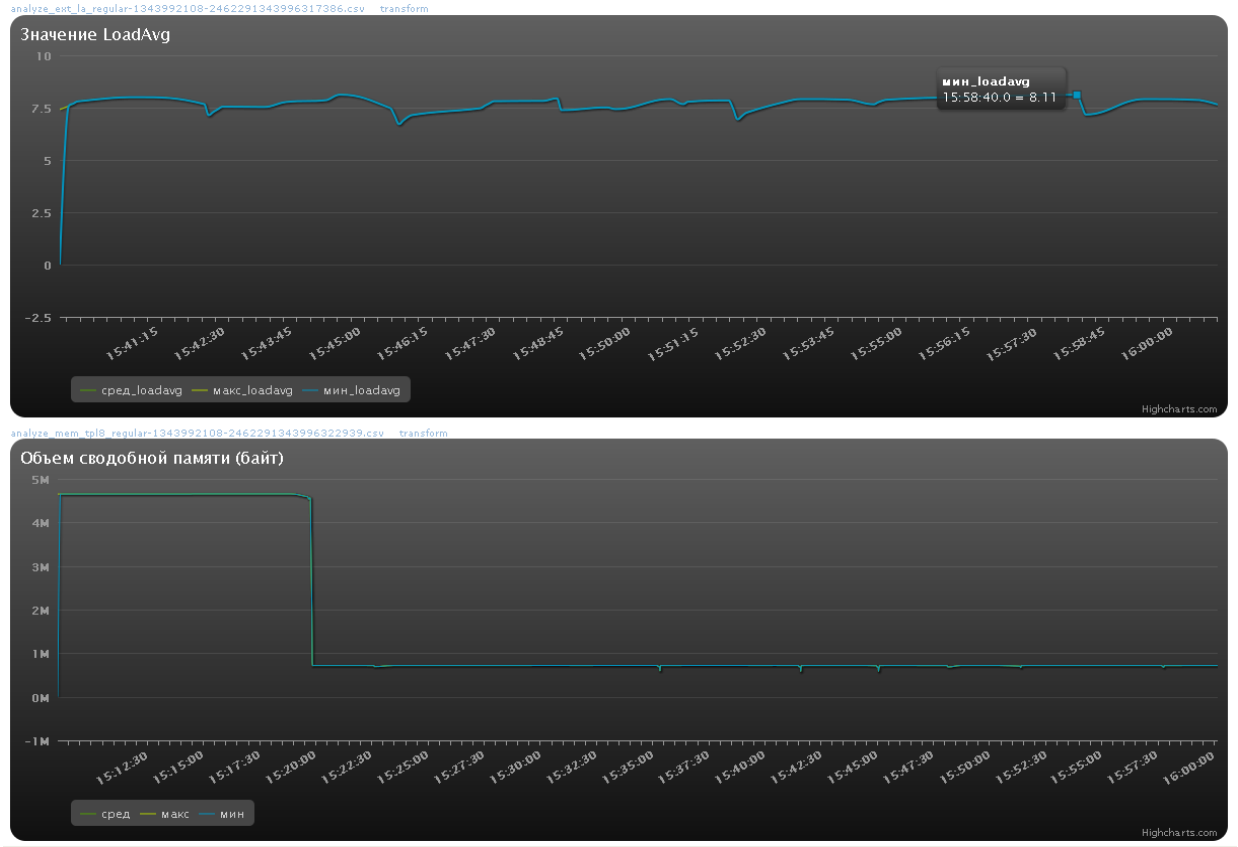

Рис. 6. Профиль использования памяти и LoadAverage

4. Заключение

Разработан и проходит апробацию удобный и эффективный инструмент для качественного анализа каждого отдельного запуска задачи, не требующий специальной подготовки задачи перед постановкой на счет. Это позволит существенно улучшить понимание пользователем того, как задача выполнялась на конкретной системе в реальных условиях. В частности, выделить аномалии, вызванные особенностями параллельной программы, или локализовать по времени или вычислительным узлам влияние других приложений. Во втором случае администратор потенциально сможет сократить это влияние, внеся соответствующие изменения в политики распределения задач или обнаружив и устранив какой-то дефект в настройке системного программного обеспечения.

Подход, безусловно, будет развиваться далее, не только совершенствуя текущий функционал, но и расширяясь качественно новыми возможностями. В частности, в подход изначально заложена идея автоматического выделения потенциально узких мест в конкретном запуске задачи, что позволит даже в автоматическом режиме обратить внимание пользователя на потенциальную проблему, а в каких-то случаях и рекомендовать инструментарий для дальнейшего изучения приложения. Автоматическая обработка данных мониторинга и выдача рекомендаций очень важны для пользователей суперкомпьютера, которые могут быть не очень знакомы с тонкостями выполнения задач на суперкомпьютерных системах, а потому самостоятельно могут не заметить какой-то характерный признак неэффективности. Для дальнейшего комплексного изучения динамических свойств хода выполнения программ в разрабатываемой системе HOPSA предусматриваются интерфейсы взаимодействия (например, через расширенную трассу OTF2) с внешними анализаторами, такими как Vampir, Scalasca и т. п.

Апробация описанного разработанного подхода проводится в Суперкомпьютерном комплексе МГУ имени М.В. Ломоносова.

Работа выполнена при финансовой поддержке Министерства образования и науки Российской Федерации, ГК № 07.514.12.4001.

ЛИТЕРАТУРА:

1. Разработка методов и инструментальных систем для анализа эффективности работы параллельных программ и суперкомпьютеров (официальный сайт российкой части совместного проекта HOPSA) - http://hopsa.parallel.ru
2. HOlistic Performance System Analysis (EU HOPSA website) - http://vi-hps.org/projects/hopsa/

3. «Hoplang — язык обработки потоков данных мониторинга» А.В. Адинец, С.А. Жуматий, Д.А. Никитенко - Сборник трудов международной научной конференции «Параллельные вычислительные технологии 2012» (ПаВТ'12), 2012, с. 351–359
4. Язык обработки потоков данных для систем кластерного мониторинга Hoplang. URL: http://github.com/zhum/hoplang