Разработка методов обнаружения неэффективной работы параллельных приложений

К.В. Иванов

ФГУП ВНИИА имени Н. Л. Духова

В работе представлен прототип разрабатываемой системы мониторинга высокопроизводительных вычислительных ресурсов. В статье описывается подход к системе мониторинга высокопроизводительных вычислительных ресурсов как к системе, ориентированной на приложения. Описывается архитектура и общая схема работы системы. Приводятся алгоритмы обнаружения неэффективной работы приложений и представлены методы задания шаблонов неэффективного поведения приложений.

1. Введение

Во многих современных научных областях всё чаще возникают задачи, требующие большое количество ресурсов [1]. Зачастую решение таких задач возможно только с использованием суперкомпьютеров. В силу разнородности вычислительных приложений, с помощью которых решаются задачи, существует большое количество архитектур современных суперкомпьютеров, которые рассчитаны на определенное множество задач. При этом эффективность выполнения задач зачастую сильно зависит от архитектуры вычислительного поля [2].

При использовании универсальных архитектур суперкомпьютеров (обладающих различными вычислительными полями) возникает задача оценки эффективности проводимых расчетов [3]. Помимо обнаружения низкой эффективности выполняемых задач, необходимо проводить анализ на возможность оптимизации работы приложения. Это необходимо не только для понимания реальной утилизации системы в целом, но и для формирования политики планирования заданий, так как эффективность выполнения параллельного приложения напрямую зависит от предоставляемых системой ресурсов. [4]

Всё это приводит к необходимости создания инструмента, который позволил бы на основе данных мониторинга о выполнении задачи на кластере сделать вывод об эффективности работы приложения и возможности его улучшения [6]. Подобный инструмент разрабатывается в Институте Автоматики им. Духова и получил название MONIKA – “MONItoring for Keen Analysis” [5].

Разрабатываемая система мониторинга обеспечивает классификацию расчетных задач и предоставляет функционал для автоматического перераспределения ресурсов в зависимости от эффективности модели поведения расчетного приложения [6]. Эффективность работы данной системы во многом зависит от качества метаданных об проводимых расчетов и наличия множества вариантов расчетных полей. Но даже при выполнении всех условий необходимо учитывать специфику выполняемых задач, и то, что в некоторых случаях использование данной системы мониторинга может быть нецелесообразно [7].

В данной статье описывается модель разрабатываемой системы мониторинга, которая ориентирована на пользовательские расчетные задачи. Разработанная модель позволяет обнаруживать неэффективные участки работы приложений. Приводятся механизмы составления и использования шаблонов для более глубокого поиска узких мест приложения. Также показаны результаты проведения тестовой эксплуатации на вычислительных кластерах института автоматики им. Духова.

2. Модель работы приложения

Общая схема работы MONIKA представлена на рис.1. Программный комплекс состоит из двух основных компонентов – система мониторинга и система анализа. Система мониторинга

имеет классическую клиент-серверную архитектуру и собирает основную информацию об утилизации вычислительных узлов. Собираются данные о следующих устройствах:

- Ядра системы
- Оперативная память (процент утилизации)
- Инфинибэнд счетчики (загрузка канала, счетчик ошибок)
- Сетевые адаптеры
- Загрузка системы ввода/вывода.

Собираемые данные поступают в систему мониторинга, которая группирует и обрабатывает результаты. Группировка может происходить по интервальному времени, что существенно уменьшает количество хранимых данных. Данная группировка предполагает разбиение всей шкалы утилизации (от 0% до 100%) на несколько диапазонов, и при наличии нескольких последовательных попаданий в один и тот же интервал активируется свертка данных значений в одно.

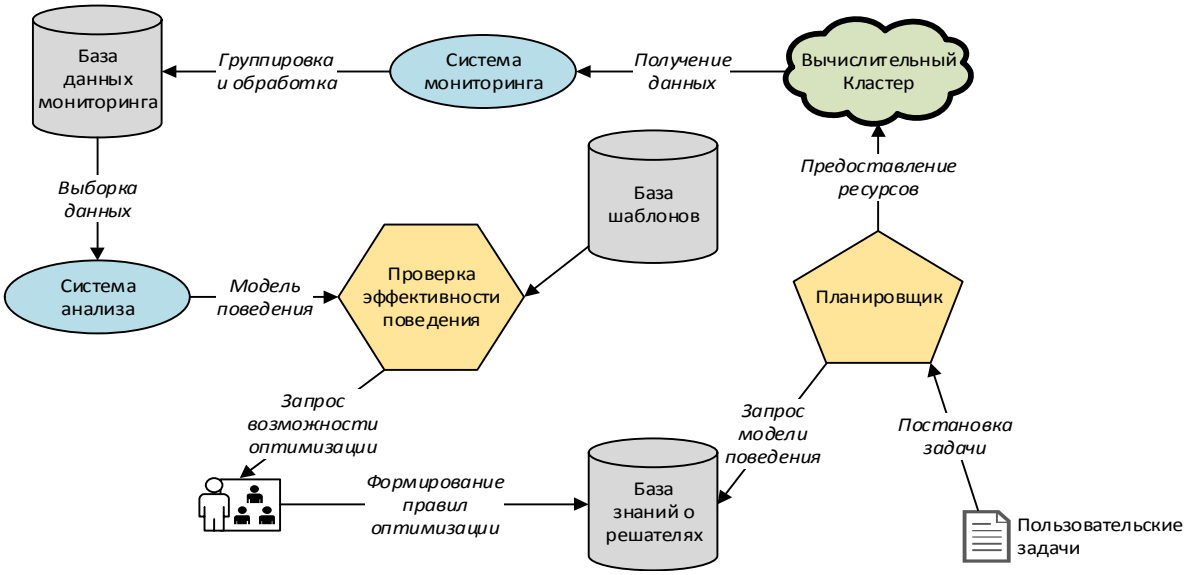

Рис 1. Общая схема работы MONIKA

Система анализа получает данные из базы данных мониторинга (или одной из её репликации) и составляет модели расчета и текущего поведения задачи. Модель расчета формируется на основе метаданных о задаче, которые получается путем анализа файла задачи и входящего файла задачи. Моделью текущего поведения задачи является статистика утилизации ресурсов, отданных под данный расчет.

Далее на основе заданных шаблонов неэффективного поведения приложений производится поиск задач с низкой эффективностью выполнения. Помимо автоматического поиска по шаблонам, предусмотрены инструменты для более глубокого, ручного анализа поведения задачи. Затем запрашивается список действий, который необходимо выполнить системе для оптимизации задачи. Данный список составляется экспертом на основе знаний о задачи. Если неэффективный участок работы приложения не удается оптимизировать, то подбираются специальные настройки приоритета выделения ресурсов данной задачи, так как чем меньше ресурсов выделяется задачи, тем ниже абсолютный простой вычислительных ресурсов. Таким образом, формируется экспертная база моделей неэффективного поведения задач.

В следующий раз, при постановке задачи в очередь на кластер, система выполняет поиск по базе знаний на предмет наличия информации о типовом поведении данного расчета. Если такая информация присутствует в базе, то планировщик выполняет действия, соответствующие найденной модели поведения.

3. Методы определения неэффективного поведения

Для определения эффективности выполнения задачи предусмотрены несколько методов:

- Статический критерий неэффективного использования ресурсов. Таким критерием является жестко заданное правило, при выполнении которого поведение расчетной задачи на кластере считается неэффективным. К таким правилам относятся правила типа: [загрузка устройства] (превышает/меньше) [значения].
- Косвенный критерий эффективности параллельности приложения, который формируется после завершения работы приложения, путем анализа использования ресурсов. Основными инструментами для такого анализа являются MPI трассировщики, такие как vampirtrace, openstk.
- Соответствие неэффективному шаблону поведения. Шаблон формируется автоматически из истории утилизации неэффективного приложения и представляет из себя множество массивов значений загрузки необходимых типов устройств. Соответствие между ходом расчета и шаблонами неэффективного поведения определяется специальными алгоритмами.

На рисунке 2 представлена визуализация утилизации центрального процессора и сетевых дисковых ресурсов. На графике ‘а’ мы можем видеть одну из типовых ситуаций неэффективного использования ресурсов – при записи на диск приложение некорректно использует CPU. Основной проблемой поиска подобных ситуаций в других приложений является невозможность формального описания поведения – так как характер поведения может повторяться при совершенно других значениях использования ресурсов (на примере – график ‘б’).

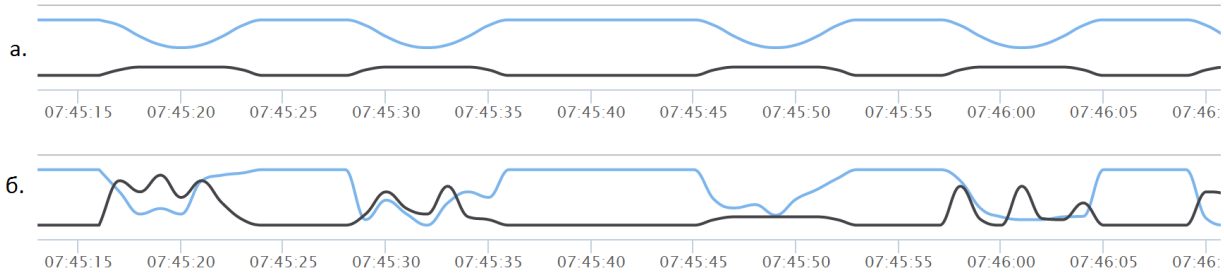

Рис 2. Утилизация CPU (синий), и диска (черный)

Для реализации механизма поиска неэффективного поведения приложения по шаблону был придуман специальный механизм для нахождения подобия. Схематично механизм представлен на рисунке 3. Таким образом искомый шаблон поведения разбивается на множество n ситуаций $A = \{ x_1, x_2, \ldots, x_n\}$, где каждая ситуация однозначно интерпретируется на графике путем проверки на соответствие выявленными критериям. Далее на основе исторических данных о шаблонной ситуации можно получить множество переходов $B = \{x_1 \rightarrow x_2, x_2 \rightarrow x_3, \ldots , x_{n-1} \rightarrow x_n\}$. Множество переходов B хранится в базе и является внутренним представлением шаблона неэффективного поведения приложения.

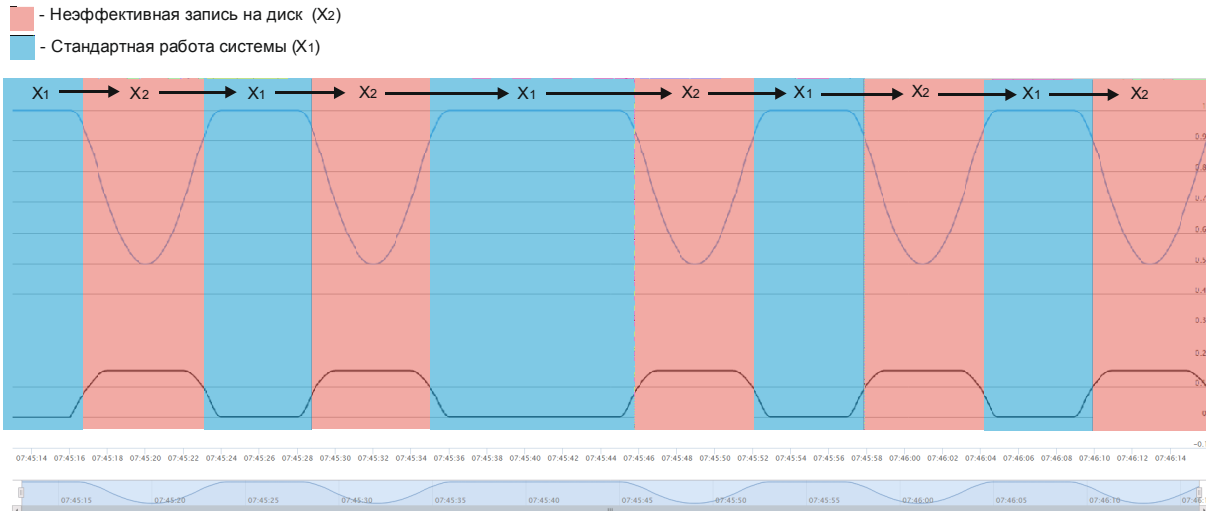

Рис 3. Механизм выявления шаблона поведения

В ходе работы последующих приложений, их поведение разбирается и формируется множество ситуаций A_n соответствующее утилизации ресурсов под данную задачу. Затем множество переходов B_n анализируется на содержание шаблонов неэффективного поведения, и, если таковые найдены, то приложение считается неэффективным.

4. Методы оптимизации вычислений

После того, как система обнаружила что приложение отработало неэффективно, необходимо сделать анализ его работы на предмет возможности оптимизации. При этом в системе рассматриваются три различных варианта оптимизации:

- Приложение возможно оптимизировать, путем изменения кода приложения. Если исходники системы открыты, то решение вопроса оптимизации напрямую зависит от действий программиста, в этом случае разработчику отправляется информация по утилизации системных ресурсов с указаниями на области, которые необходимо оптимизировать.
- Приложение возможно оптимизировать, путем перераспределения ресурсов. Данный метод применяется, если приложение работает неэффективно за счет недостаточной производительности того или иного ресурса. К таким ситуациям можно отнести перенос приложения на параллельную файловую систему, узлы с большой памятью, дополнительные совычислители или с другой сетевой топологией.
- Приложение невозможно оптимизировать. К таким приложениям можно отнести коммерческие продукты, которые не поддаются тонкой настройке. Единственным способом повышения утилизации системы является понижение приоритета выдачи ресурсов задачам такого класса.

За счет описанных методов оптимизации возможно существенно повысить процент утилизации вычислительных ресурсов. Однако, повышение сильно зависит не только от гибкости архитектуры кластера, но и от политик управления очередью задач, и, если в системе имеются строгие правила ограничений по ресурсам, то система теряет свою эффективность.

5. Заключение

В настоящее время проводиться тестовая эксплуатация системы MONIKA на вычислительных ресурсах ВНИИА. Использующиеся в тестах кластеры обладают хорошими условиями для функционирования системы – имеется большое количество разнородных вычислительных полей, и они обладают относительной свободой в оптимизации алгоритмов планировщиков.

В результате эксплуатации были выявлены некоторые ошибки в разрабатываемых приложениях. Также была повышена утилизация ресурсов за счет изменения политик планировщика, путем уменьшения приоритета низкоэффективных типов решателей.

В дальнейшем планируется исследование различных алгоритмов на поиск шаблонов в ходе работы приложения и дополнение системы мониторинга информацией об использовании инфраструктуры вычислительных комплексов (таких как температура и энергопотребления) для исследования влияния расчетных задач на эффективность потребления электричества.

Литература

1. Хорошевский В.Г. Архитектура вычислительных систем. – М.: МГТУ им. Н.Э. Баумана, 2008. – 520 с.
2. Новиков А, Б., Петунин С. А. Влияние специализированных алгоритмов планирования заданий на эффективность использования вычислительных ресурсов в частных случаях // XIII международный семинар "Супервычисления и математическое моделирование": Тез. доклада РФЯЦ-ВНИИЭФ, 2011г. 17 с.
3. 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", Изд-во МГУ Москва, 2012. с. 9
4. Иванов К. В. Система мониторинга точек предельной нагрузки в кластерных вычислениях. “Вестник УГАТУ, Том 18, № 3(64) (2014), 290-294 с.
5. Иванов К. В. Система мониторинга с прогнозированием ошибок. Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции (1–5 апреля 2013 г., г. Челябинск). Челябинск: Издательский центр ЮУрГУ, 2013. 637 с.
6. Головинский А. Л., Белоус Л. Ф., Маленко А. Л. Веб-портал системы управления суперкомпьютером // Вычислительные методы и программирование. 2010. 11/7.
7. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб.: БХВ-Петербург, 2002. — 608 с.