АДАПТАЦИЯ СИСТЕМЫ ВЕДЕНИЯ ЗАДАЧ ПОЛЬЗОВАТЕЛЕЙ SLURM С ЦЕЛЬЮ УЧЁТА ЛИЦЕНЗИЙ ANSYS

А.Н. Сальников[1,2], В.Д. Никоноров[2], П.И. Каледа[2]

[1] факультет ВМК МГУ имени М.В. Ломоносова
[2] ОАО НИКИЭТ

Высокопроизводительные вычисления с использованием современных коммерческих пакетов программ требуют учёта лицензий и, как следствие, работы с сервером лицензий. К таким пакетам можно отнести Ansys, FlowVision, Star-CD. Нормальная политика системных администраторов в этой ситуации — закрепить определённое число лицензий за кластером и в дальнейшем средствами систем ведения очередей и алгоритмами планирования заданий учитывать соответствующие лицензии. Однако такой подход не всегда возможен по причинам денежно-финансового характера. На практике одни и те же лицензии могут использоваться как на узлах вычислительного кластера, так и на рабочих станциях пользователей при проведении отладки расчётных приложений.

Как правило, лицензии для приложений управляются при помощи специального сервера лицензий. Довольно популярным таким сервером является FlexLM (в текущий момент FlexNet [1]). Это приводит к трудностям в планировании задач пользователей, так как система ведения очередей в общем случае не может контролировать доступность лицензий в тот момент, когда построено расписание выполнения задач.

Некоторые системы ведения очередей позволяют работать с серверами лицензий. К таким системам относятся: Moab, Sun Grid Engine, IBM Platform License Sceduler. Однако не на всех кластерах используются именно эти системы. Кроме того, серверов управления лицензиями может быть много, и ожидать, что все они одновременно будут поддержаны одной конкретной системой ведения очередей - не приходится.

В данной работе мы остановились на системе ведения очередей Slurm. Такой выбор был обусловлен следующими причинами. В ОАО «НИКИЭТ» кластеры управляются системой ведения очередей Slurm. Важно так же, что Slurm — довольно популярная система с открытым исходным кодом.

В ОАО «НИКИЭТ» многие пользователи для расчётов используют коммерческий пакет Ansys, в том числе на вычислительном кластере. При этом лицензии используются как на вычислительном кластере, так и на рабочих станциях пользователей. Все они управляются с одного и того же сервера лицензий. Возникает проблема: когда пользователь поставил задачу в очередь, необходимые лицензии были доступны, а когда задача начала исполняться лицензии оказались заняты кем-то другим. С целью минимизации числа снятий задачи со счёта из-за проблем доступности лицензий были предприняты следующие действия:

1. Набор лицензий «anshpc_pack» (HPC лицензии) закреплён за кластером. Эти лицензии используются приложениями Ansys в том случае, когда программа запускается в многопроцессорном режиме. При этом число лицензий, которое используется, вычисляется в зависимости от числа задействованных процессоров нелинейным образом. Формула будет обсуждена позже.
2. Создан Python скрипт, который позволяет указать необходимые лицензии, отличные от anshpc_pack. Этот скрипт вычисляет также, в зависимости от числа задействованных Ansys-ом MPI-процессов, число лицензий anshpc_pack.
3. Исходный код сервера slurmctld был модифицирован таким образом, чтобы он через определённые промежутки времени опрашивал сервер лицензий и обновлял по информации, полученной с него, состояние внутреннего списка лицензий

Теперь подробнее про каждое действие. Первое действие производится легко, для этого было достаточно указать в конфигурационном файле список машин, с которых разрешается использовать лицензию (по умолчанию сервер лицензий позволяет захватывать лицензию любому подключившемуся). В этот список внесли все узлы кластера. В результате никто не может себе захватить HPC лицензию на рабочую станцию. Кроме всего прочего это позволяет дисциплинировать пользователя, чтобы все длительные расчёты велись пользователями на вычислительном кластере организации.

Для второго действия пришлось учитывать формулу, по которой вычисляются HPC лицензии [2]. В Python скрипте была применена следующая формула:

$$lic = \lceil min\left(1, \left(\log_2(N \cdot p) - 1\right)/2\right) \rceil$$

Здесь N — число задействованных при постановке задачи в очередь узлов кластера, а p — число MPI-процессов, которое будет приходиться на один узел. Скрипт написан так, что его можно предложить пользователю использовать как замену стандартной утилиты Slurm sbatch. Данный скрипт поддерживает набор наиболее важных параметров sbatch. Он по числу узлов и процессов на узел определяет число лицензий HPC и указывает это число лицензий как ресурс задачи при постановке в очередь. Самостоятельно ставит задачу в очередь, формируя BASH-скрипт для запуска и запуская внутри себя команду sbatch. Таким образом,

пользователю нет необходимости самостоятельно высчитывать число HPC лицензий при постановке задачи в очередь.

Авторами было произведено предварительное исследование того, как именно лицензии Ansys используются в организации. С этой целью написан специальный BASH-скрипт, который через каждые 15 минут опрашивает сервер лицензий и собирает статистику их использования. Полученные данные были обработаны, а именно была извлечена информация о некоторых наиболее интересных типах лицензий:

1. **anshpc_pack —** лицензия использующаяся при запуске MPI-реализации «решателя» Ansys. Данный вид лицензий обсуждался в тексте ранее.
2. **acfd** — лицензия позволяющая запуск клиентского приложения для работы с данными об объекте через графический интерфейс.
3. **acfd_solver** — лицензия для проведения расчётов по газодинамике, которая захватывается программой непосредственно в момент счёта, независимо от того, параллельная или последовательная версия решателя используется.
4. **agppi** — лицензия использующаяся при создании и редактировании 3-х мерной модели области, в которой в дальнейшем будет производиться расчёт.
5. **preppost** — лицензия, позволяющая описывать и редактировать структуру объекта состоящего из композитных материалов.

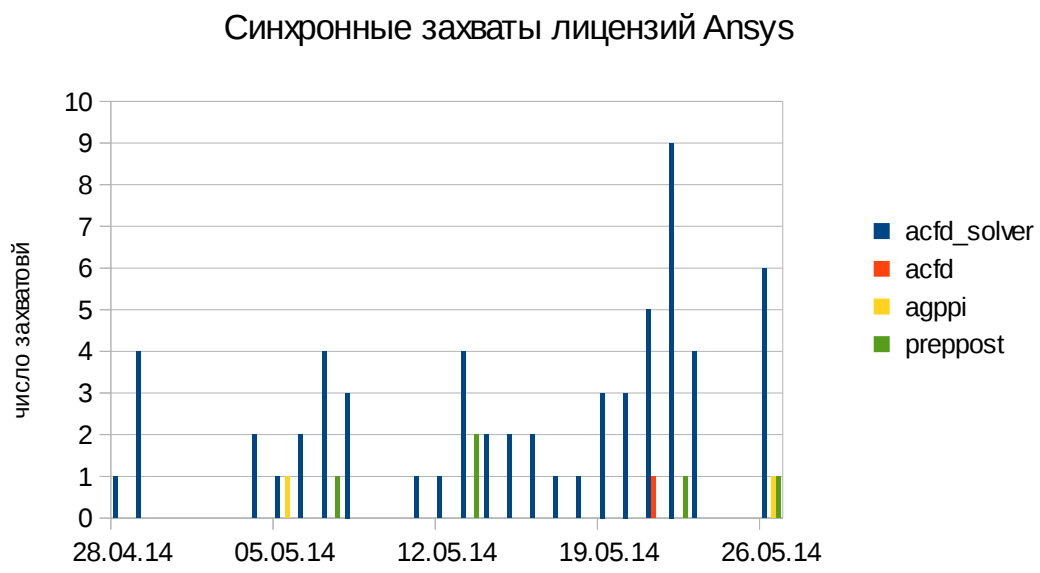

Рис. 1. На диаграмме отображено число синхронных захватов лицензий с лицензией anshpc_pack.

Было выяснено, что пользователи запускают решатели на своих рабочих станциях достаточно редко, что запуски приложений, которые представляют собой графический интерфейс для постобработки и визуализации результатов расчётов, практически никогда не происходят одновременно с запуском расчёта на кластере. На рисунке 1 представлена диаграмма, где каждый столбец задаёт число одновременных захватов лицензий HPC и лицензий acfd, acfd_solver, agppi, preppost за сутки в течении месяца. Видно, что acfd_solver используется совместно с HPC лицензиями намного чаще, чем лицензии, связанные с визуализацией и построением области расчёта. Это позволило сделать вывод, что ситуация захвата решателя с рабочей станции пользователя будет редка и возможен подход, когда задача откладывается в очереди из-за нехватки динамически захватываемых с точки зрения кластера лицензий.

Было принято решение модифицировать исходный код slurmctld так, чтобы он мог учитывать динамически изменяемые лицензии. FlexLM предоставляет консольную программу «lmutil». С помощью этой программы можно запросить информацию об использовании лицензий сервером лицензий. Было решено запускать эту программу через определённый промежуток времени, собирать сведения о использующихся в текущий момент лицензиях и обновлять информацию о лицензиях в системе ведения очередей Slurm. Был выбран интервал времени, за который вероятность захвата лицензии, так чтобы Slurm об этом не узнал, была минимальна, но при этом опрос сервера лицензий не приводил бы к существенному увеличению нагрузки как на сервер лицензий, так и на систему ведения очередей.

Для дальнейшего понимания способа изменения исходного кода Slurm в общих чертах опишем архитектуру программного кода программы slurmctld. Данная программа является сервером, который отслеживает состояние узлов кластера посредством обращения по сети к «демонам», управляющим непосредственно запуском и остановкой задач пользователей на узлах кластера. Так же он отслеживает и

управляет очередью заданий, в том числе осуществляет планирование запусков заданий. Для планированя запусков могут использоваться различные алгоритмы, наиболее популярный из них Backfill (Алгоритм обратного заполнения очереди).

Программный код slurmctld является многопоточным приложением с набором внутренних библиотечных функций (к сожалению, архитектура библиотеки и функции библиотеки не описаны в документации к Slurm). В slurmctld есть механизм plugin-ов, данный механизм позволяет написать программный код, который использует внутреннюю библиотеку функций для достижения своих целей. Например, можно создать реализацию своего собственного алгоритма планирования задач. Все plugin-ы разделены на два множества. Множество принудительно загружаемых plugin-ов, они не могут быть выгружены, пока slurmctld работает, и множество plugin-ов, подгружаемых по запросу. Например, есть plugin sched — который обеспечивает функции для работы с очередью, и есть plugin backfill — который использует функции предыдущего plugin-a sched для непосредственного управления очередью. Код backfill может быть заменён кодом другого алгоритма, а код sched не может быть заменён. Все plugin-ы в Slurm оформлены как отдельная нить в терминах библиотеки pthread.

В Slurm можно зарегистрировать лицензию на программное обеспечение как ресурс. В дальнейшем этот ресурс учитывается при планировании заданий. Например, в реализации алгоритма backfill запуск задачи на счёт откладывается, при этом сохраняется место задачи в очереди. В программном коде slurmctld информация о лицензиях хранится в виде списка, в котором хранится имя лицензии, число использованных лицензий, общее число доступных лицензий. Задача при постановке в очередь указывает, сколько лицензий с определённым именем ей необходимо для работы; при запуске задачи на выполнение добавляется значение в число использованных лицензий. Сам список является критическим ресурсом и защищён семафором.

В процессе поиска места, в которое можно встроить исходный код, выяснилось следующее. Механизм запуска остановки plugin-ов в Slurm не столь очевиден, как может показаться с первого взгляда. Написание plugin-а, который изменяет список лицензий, невозможно без изменения кода остальной части slurmctld. В результате было принято решение о написании кода, который максимально по поведению и системе регистрации себя в Slurm походил на plugin, но запускался бы одновременно с backfill, используя одни и те же средства запуска, что и для backfill.

Итак, созданный plugin, условно «dynamic_licensing» - представляет собой поток в терминах pthread. Он через определённые промежутки времени создаёт новый процесс, в данном процессе запускает программу lmutil с нужными параметрами, причем вывод этой команды передаётся обратно потоку. Поток разбирает вывод от lmutil и в соотвествии с этим формирует внутренний список информации о лицензиях. В этом списке к имени каждой лицензии приписывается префикс «dynamic_», чтобы они не путались с обычными статическими лицензиями, которые указываются в конфигурационном файле Slurm (slurm.conf). После этого семафор, связанный с глобальным списком лицензий в slurmctl, блокируется, и сам список обновляется.

Таким образом, пользователь при постановке задачи в очередь может указать лицензию, которая отсутствует в конфигурационном файле Slurm и может появится или исчезнуть в процессе работы slurmctld. При этом реализуется корректный механизм откладывания запуска задачи, если лицензия не доступна.

Предложенный подход не гарантирует, что задача не встанет на счёт в случае отсутствия лицензии на сервере лицензий. Однако, подбирая частоту опроса сервера лицензий, можно свести вероятность отказа запуска к минимуму. Опыт использования в ООО «НИКИЭТ» показал жизненность предлагаемого подхода.

ЛИТЕРАТУРА:

1. Сайт компании Flexera software http://www.flexerasoftware.com/products/entitlement-management/flexnet-licensing/ [Дата обращения 19.05.14].
2. Сайт российского представительства компании Ansys: страница посвященная учёту лицензий для высокопроизводительных вычислений: http://www.cadfem-cis.ru/products/ansys/workflow/hp/packs/ [Дата обращения 26.05.14].