Подсистема автоматического распределения вычислительных объектов по процессорам

Станислав Ворошилов

Научный руководитель: Илюшин А.И.

Механико-математический факультет
Кафедра вычислительной механики
19 Мая, 2014 г.

Цель работы

Цель

Создание подсистемы подкачки объектов в системе OST.

Intel MIC

Работа происходит на многоядерной процессорной системе **Intel Xeon Phi** (*архитектура MIC*) - сопроцессорная плата-ускоритель. Аналогично графическим усорителям здесь используются ускорители типа Pentium (240 шт.)

Цель работы

Цель

Создание подсистемы подкачки объектов в системе OST.

Intel MIC

Работа происходит на многоядерной процессорной системе **Intel Xeon Phi** (*архитектура MIC*) - сопроцессорная плата-ускоритель. Аналогично графическим усорителям здесь используются ускорители типа Pentium (240 шт.)

Запуск задачи в системе OST

- В качестве главной программы выступает программа OST
- Представление подобласти - **программный объект**: набор подпрограмм + данные
- Модель - набор объектов в файле объектов
- Двухуровневая среда для хранения и счета объектов:
  - обычный файл для хранения модели до счета и после контрольных точек
  - набор процессорных узлов (реальных или виртуальных), выделенных для счета модели

Запуск задачи в системе OST

- Раньше: простейший вид - число объектов No равно числу процессоров Np
- Однако, если $No > Np$ (что имеет существенный практический смысл в организации счета прикладных задач), то необходимо динамически распределять объекты между процессорами

Основные проблемы

Прикладной программист:

- Долгое время ожидания в очереди на необходимые ресурсы (заказанное число процессоров)

Администратор МВС:

- Запуск высокоприоритетных задач вне очереди
- Минимизация простоев процессоров в МВС

Основная идея решения

статически до начала счета распределять виртуальные процессоры, а отображать каждый виртуальный процессор на реальный – динамически во время счета

Поэтому $No = Np_{virtual}$

Основные проблемы

Прикладной программист:

- Долгое время ожидания в очереди на необходимые ресурсы (заказанное число процессоров)

Администратор МВС:

- Запуск высокоприоритетных задач вне очереди
- Минимизация простоев процессоров в МВС

Основная идея решения

статически до начала счета распределять виртуальные процессоры, а отображать каждый виртуальный процессор на реальный – динамически во время счета
Поэтому $No = Np_{virtual}$

Аналогия с виртуальной памятью

Объект достаточно сложная структура, однако в поставленной задаче он является аналогом страницы в виртуальной памяти. По сути нам необходимо сделать отображение множества объектов на множество процессоров, но так как объектов гораздо больше процессоров, то нужно вводить виртуальные процессоры, которые позволят сделать требуемое взаимооднозначное соответствие.

Модель данных

Logical reference

Логическим обращением к объекту p называется выполнение какой-либо операции над данными, содержащимися в объекте p. **(logical reference - LR)**

Physical reference

Физическим обращением к объекту p называется логическое обращение к этому объекту, приводящее к его считыванию с внешнего устройства хранения данных. **(physical reference - PR)**

Модель данных

Logical reference

Логическим обращением к объекту p называется выполнение какой-либо операции над данными, содержащимися в объекте p. **(logical reference - LR)**

Physical reference

Физическим обращением к объекту p называется логическое обращение к этому объекту, приводящее к его считыванию с внешнего устройства хранения данных. **(physical reference - PR)**

Модель данных

Вывод:

В контексте введенной терминологии задачей алгоритма замещения является минимизация отношения числа PR к LR для заданной последовательности обращений к блокам данных.

Алгоритмы подкачки

Demand paging algorithms

При необходимости замещения среди объектов один единственный и замещается

Prepaging algorithms

Для замещения выбираются сразу несколько объектов. Алгоритмы данного типа могут быть эффективны в особых случаях, но потенциально сложны в реализации.

В работе применялся **demand paging** алгоритм

Проблемы при подкачке

Основной проблемой при подкачке объектов является так называемый ***trashing*** - аналогичная ситуация в виртуальной памяти. Пусть у нас есть объект, сделавший вызов на объект, которого нет на узле. Данный объект подкачивается, для чего выталкивается первый объект, после чего все происходит наоборот.

Обработка вызова

Ключевая функция планировщика ***ObjectFault*** работает следующим образом:

- Вызов операции по ссылке на объект идет по физическому адресу (из этой ссылки) реального процессора, в котором находится вызываемый объект. Если физ. адрес пуст или объект в данном процессоре отсутствует (ранее вытолкнут) то
- Планировщик идет в файл объектов и находит там нужный объект.

Обработка вызова

- Планировщик ищет свободный процессор или, если такового нет, ищет объект кандидат на выталкивание. Производится выталкивание объекта в файл объектов.
- Вызываемый объект вталкивается на свободный или освобожденный процессор. В планировщик возвращается его новый физ. адрес и он записывает его в ссылку на объект.
- Повтор вызова

Реализация

Реализация проводилась на Intel Xeon Phi (MIC)

Среда

Операционная и программная среда состоит из операвционной сиситемы Linux на каждом процессоре с компиляторами Fortran, C++ поддерживающими многопроцессорую работу. MPI для связи процессоров

Python

К сожалению, компанией Intel не предусмотрена поддержка языка Python на котром написана система OST. Необходимым шагом была установка Python'а и основных библиотек необходимых для работы и расчетов

Реализация

Система OST

Необходимо перенести систему на новую платформу с учетом внутренних особенностей MIC

Подкачки

Собственно реализация алгоритма подкачек и проведение тестов

Сайт проекта

Для поддержки проекта и удобства ознакомления с представляемой системой было решено создать сайт где размещалась бы вся актуальная информация по работе на плате Intel Xeon Phi и в системе OST

Результаты

Среда

Была установлена операционная система *Linux 2.6.32-358.el6.x86_64*, установлены пакеты Intel ComposerXE включающие компиляторы Fortran и C++, поддерживающие многопроцессорную работу. Установлена поддержка MPI

Тесты

Успешно проведены тестово-производственные счеты сотрудниками ИПМ (В.Жуков, Н.Новикова, М.Краснов)

Результаты

OST

Была перенесена на MIC версия системы OST и проведены успешные тесты

Подкачки

Запрограммирована первая версия с подкачками и пропущены первые тесты

Сайт проекта

Запущен сайт по адресу ***http://ost.kiam.ru/*** на котором можно найти необходимую информацию по работе с системой OST и Intel Xeon Phi