Политика регулирования частот центрального процессора в ядре Linux для приложений реального времени

Глаз Роман Сергеевич

Научный руководитель:
Кринов Пётр Сергеевич, к. ф.-м. н.

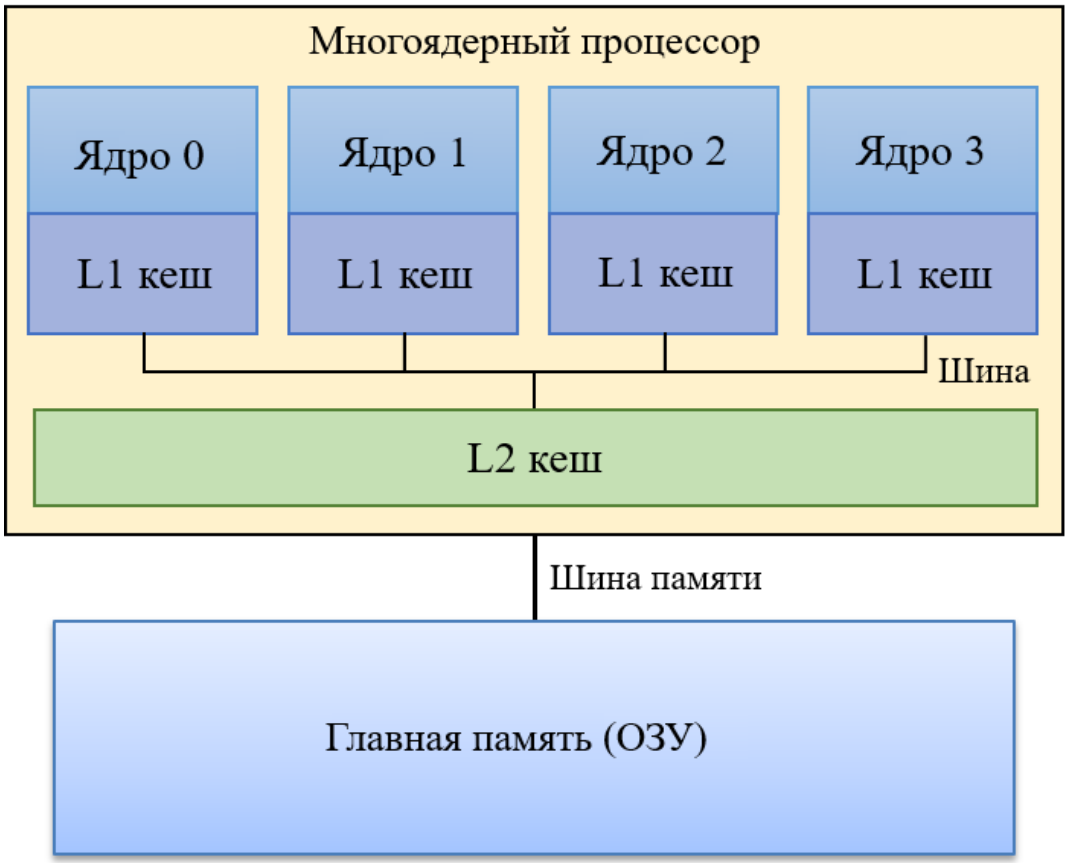
Выпускная квалификационная работа бакалавра
7 июня 2024 г.

Содержание

Актуальность

1. Производительность современных ЦП ограничена скоростью системы памяти.
2. Память: несколько уровней кешей и ОЗУ.
3. Современные ОС (Linux, FreeBSD) не учитывают задержки памяти при регулировании частот процессора и планировании задач.
4. Для приложений реального времени производительность и энергопотребление ЦП являются ключевыми факторами.
5. Gem5 – эмулятор компьютерных архитектур с открытым исходным кодом, позволяющий запускать ОС Linux.

Пример организации памяти



Пример организации памяти в многоядерном устройстве

Цели работы

1. Разработать механизмыдля сбора статистики микроархитектурных событий ядер ЦП (центрального процессора) при различных тактовых частотах для устройств эмулятора Gem5.
2. Разработать модель производительности ядер ЦП с учётом поведения и характеристик устройств памяти (кеши, ОЗУ).
3. Реализовать политику регулирования частот ядер ЦП в ОС Linux в пространстве ядра, используя разработанную модель.

Задачи

1. Реализовать в операционной системе Linux (версии 6.1):
   - 1.1 Набор драйверов, позволяющих изменять тактовые частоты ЦП в эмулируемой системе архитектурным симулятором Gem5.
   - 1.2 Возможность сбора статистики микроархитектурных событий ядер ЦП, связанных с работой кешей, для архитектуры ARM64.
2. Разработать модель производительности ядер ЦП:
   - 2.1 Спроектировать теоретическую модель производительности ЦП.
   - 2.2 Реализовать набор бенчмарков, подходящих для построенной модели, и собрать для них статистику микроархитектурных событий при различных частотах ЦП.
   - 2.3 Найти коэффициенты модели на основе снятых данных.
3. Реализовать политику регулирования частот ядер ЦП в ОС Linux в пространстве ядра, используя разработанную модель.

Обзор существующих решений

1. В ядре ОС Linux неявным образом предполагается независимость производительности ядра ЦП:
   - 1.1 В подсчёте нагрузки ядра процессора.
   - 1.2 В регулировании частот ядер.
2. Существует ряд моделей производительности ЦП, которые, как правило:
   - 2.1 Используют данные, которые нельзя получить в режиме реального времени.
   - 2.2 Не учитывают наличие возможности регулирования тактовых частот ЦП.
   - 2.3 Применимы только для конкретных приложений (отсутствует обобщение).

Исследование и построение решения

Построена следующая теоретическая модель производительности (для системы с 3-мя уровнями кешей):

$$cpi = cpi_{L1/L2} + \alpha_{mem}^{par} \cdot (\beta_1 \cdot (l2pi - l3pi) + \beta_2 \cdot l3pi) \cdot freq_{CPU} \quad (1)$$

1. cpi – количество затраченных тактов на инструкцию.
2. $l2pi$ и $l3pi$ характеризуют количество промахов в кеши 2-ого и 3-ого уровней (с некоторыми оговорками).
3. $freq_{CPU}$ – тактовая частота ядра процессора.
4. α_{mem}^{par} – параметр, характеризующий параллелизм обращений ядра ЦП в память, β_1 и β_2 – параметры системы памяти.
5. $cpi_{L1/L2}$ – количество затраченных тактов на инструкцию, если не учитывать обращения за пределами кеша 2-ого уровня.

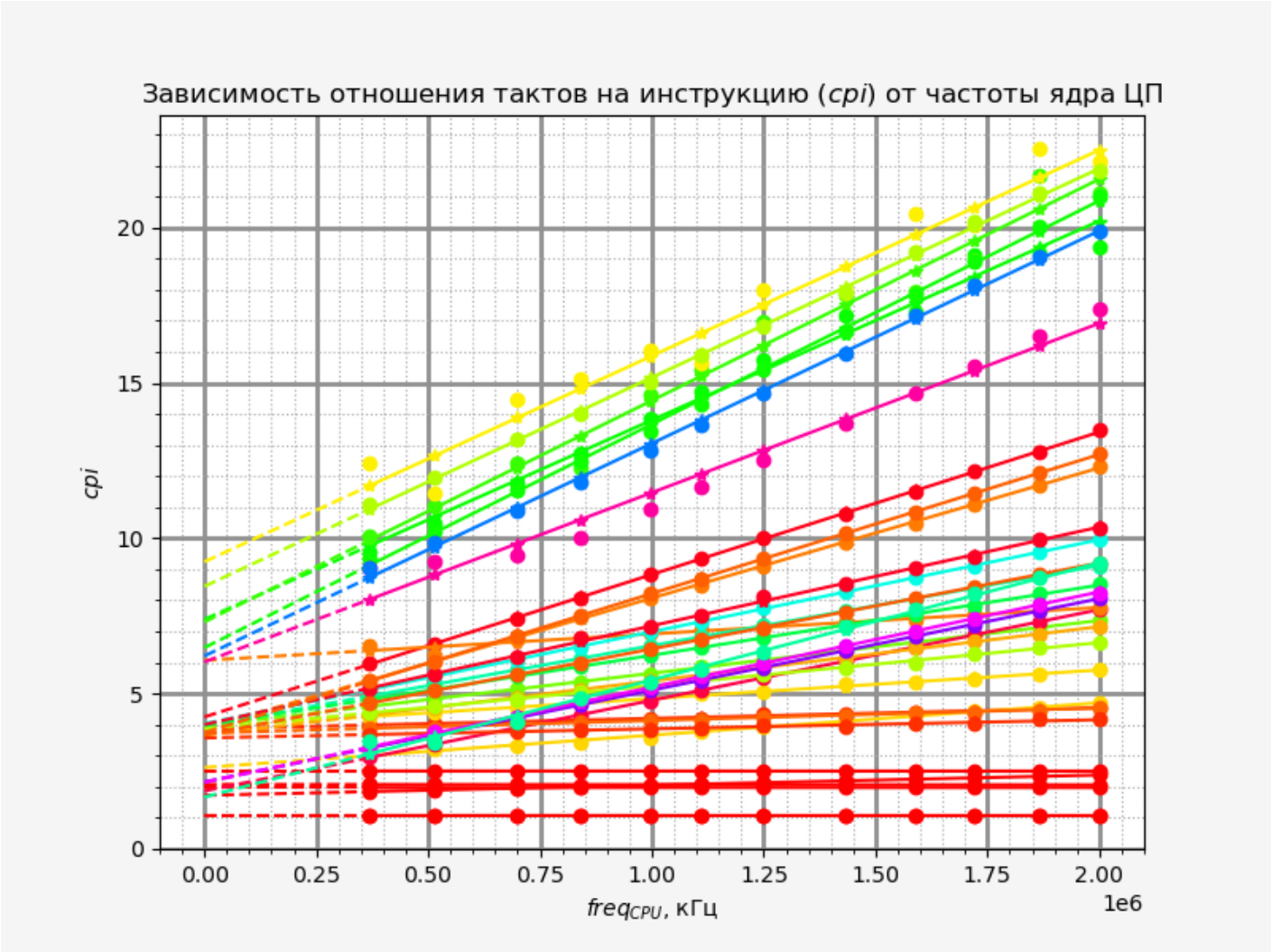
Применение теоретической модели

Предлагаемое использование модели:

1. β_1, $\beta_2 = const$, а $l2pi$, $l3pi$ являются параметрами исполняемой рабочей нагрузки, измеряются в режиме реального времени.
2. Коэффициент α_{mem}^{par} зависит от исполняемой рабочей нагрузки, и регулируется в режиме реального времени на основе ошибок предсказания.
3. Величина $cpi_{L1/L2}$ находится из всех остальных величин по формуле (зависит от исполняемой рабочей нагрузки).

Таким образом, на основе текущей нагрузки на ядро ЦП может быть выбрана минимальная тактовая частота для требуемого уровня производительности.

Сбор и обработка данных



Механизмы, реализованные в ядре Linux

1. Драйверы *clk* и *cpufreq* для поддержки изменения частот устройств Gem5.
2. Сбор статистики микроархитектурных событий для каждого ядра ЦП и каждой нити исполнения.
   - 2.1 При смене контекста нити исполнения, ротации счётчиков микроархитектурных событий и по таймеру.
3. Выбор тактовых частот на основе собираемой статистики (как по ядрам ЦП, так и по нитям исполнения, выполняемых на эти ядрах).
   - 3.1 При пробуждениях нити исполнения, при миграциях нитей исполнения на другие ядра ЦП и по таймеру.
4. Политику выставления частот на основе голосования выбранных тактовых частот по ядрам ЦП.

Результаты

Преимущества модели:

1. Учёт параметров, зависимых от рабочей нагрузки (кол-во обращений в ОЗУ, высокие уровни кешей).
2. Введение параметра параллелизма обращений в память.
3. Независимость модели от специфики работы конвейера ядра ЦП.
4. Возможность использования для многопоточных ядер ЦП.

Производительность приложений реального времени ускоряется до 5-10%.

Благодарю за внимание!