***29. Модель динамической перекомпиляции. Особенности JIT компилятора. Замена кода.***

Не нашел в презентации

Википедия:

<https://en.wikipedia.org/wiki/Dynamic_recompilation>



* Мертвый код
  + Не исполняется никогда
* Холодный код
  + Исполняется редко, не оказывает влияния на скорость работы системы
* Теплый код
  + Исполняется регулярно, влияет на скорость работы системы
* Горячий код
  + Оказывает доминирующее влияние на скорость работы системы

Семплирование:

* Остановка пользовательского кода в случайные моменты времени и анализ стека вызовов
* Почти не влияет на производительность пользовательского кода
* Является вероятностным методом, имеет сравнительно низкую чувствительность
* Для определения момента остановки может использоваться таймер или отладочное событие (переполнение счетчика инструкций, промахов кэша)
* Для остановки часто используются безопасные точки
* Используется для обнаружения горячего кода

http://www.ibm.com/developerworks/ru/java/library/j-jtp12214/

Code path компилируется в машинный код после интерпретации его определенное количество раз. Но JVM продолжает анализ и может перекомпилировать код заново с более высоким уровнем оптимизации, если решит, что code path является особенно "горячим" или последующий анализ данных показал возможность дополнительной оптимизации. JVM может перекомпилировать одни и те же байткоды много раз при выполнении одиночного приложения.

*Замена кода*

* Ленивая замена кода
  + Новый код исполняется при следующем вызове
  + Требует сохранения предыдущей версии кода если она исполняется в момент завершения компиляции
  + Требуется отслеживать ссылки на старую версию кода для освобождения памяти
  + В случае глубокой подстановки может привести к существенным потерям производительности из-за невозможности заменить долго исполняемый код
* Замена кода на стеке
  + Версия кода заменяется в момент завершения компиляции
  + Требует трансформации параметров на стеке в соответствие с новой версией кода
  + Сложность реализации