Системы управления памятью в JVM

Николай Иготти

Требования к системе управления памятью в Java

- Стандарт не специфицирует алгоритмы управления памятью, но налагает требования:
  - *Память всегда выделяется под объект (явно или неявно)*
  - *Память* **всегда** *инициализируется в значения по умолчанию*
  - *Для завершения инициализации должен вызывться конструктор*
  - *Память (возможно) освобождается при недоступности объекта, weak references*
  - *Не допускается алиасинг (перекрытие памяти объектов)*
  - *equals() и hashCode()*
  - *При недостатке памяти бросается исключение* **java.lang.OutOfMemoryError** *и его можно поймать*
  - *При освобождении объекта вызывается* **protected void finalize() throws Throwable**

Фазы жизни объекта

- Участок в хипе
- Память выделенная аллокатором (заполнена нулями)
- Инициализированный объект
- Недостижимый объект (а как с классами?)
- Финализируемый объект
- Фантомный объект
- Участок в хипе

Достижимость

- Rootset – корневые объекты
  - *Объекты на стеке Ява*
  - *Глобальные переменные Ява (какие?)*
  - *Структуры данных ВМ*
  - *Регистры процессора содержащие ссылки*
  - *Ещё?*
- Достижимо транзитивное замыкание корневых объектов по отношению “иметь ссылку”
- Собирать можно (а всегда ли нужно?) дополнение множества достижимых объектов - *мусор*

Слабые ссылки, финализация, хэш-код

- Типы ссылок:
  - *Сильная (обычная ссылка на объект* *foo.bar*)
  - *Мягкая (soft)* *java.lang.ref.SoftReference*
  - *Слабая (weak)* *java.lang.ref.WeakReference*
  - *Фантомная (phantom)* *java.lang.ref.PhantomReference*
- *java.lang.Object.finalize()*
- Разные политики обработки коллектором, интерфейс между коллектором и Ява-кодом
- Стабильность хэшкода

Факторы учитываемые при создании коллектора

- Требования по объёму данных
- Требования по производительности (с какой скоростью может создаваться мусор)
- Требования по времени жизни приложения (серверные программы могут работать годами)
- Требования по гарантии на время отклика (мягкие, жёсткие)
- Ожидаемые сценарии нагрузки
- Ожидаемое аппаратное обеспечение

Информация о типичном поведении

- Анализ реальных систем показал:
  - *Большинство объектов “умирают молодыми” (generational hypothesis)*
  - *Объекты делятся на несколько категорий с различным временем жизни*
  - *Потоки используют, в основном, данные которые сами создают*
  - *Немногие объекты используются для синхронизации*
  - *Немногие объекты хранятся в хэшах (а следовательно, требуют стабильного хэш-кода)*
  - *Взаимоотносящиеся объекты ссылаются друг на друга*

Реализации коллектора (китайская классификация)

- ~~Подсчёт ссылок, удаление при 0~~
- Точные/приближённые
- Копирование живых объектов (semi-space)
- Пометить-вычистить
- Останавливающие/инкрементальные
- Реального времени
- С учётом гипотезы поколений
- С учётом потоковой локальности
- Комбинированные

Вопросы

- Стековые переменные – это явное или неявное освобождение памяти?
- Какие проблемы и преимущества внёс бы в Яву алиасинг?
- Как влияет поддержка мягких и слабых ссылок на дизайн коллектора?
- Всегда ли наблюдается “типичное” поведение? Что, если нет?