УДК 004.032.24

В.В. Боровик
Донецкий национальный технический университет
кафедра автоматизированных систем управления

JAVA – РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Аннотация

Боровик В.В. Java – реализация параллельного генетического алгоритма .*Рассмотрена концепция параллельного генетического алгоритма. Определены проблемы построения параллельного генетического алгоритма. Рассмотрена реализация параллельного генетического алгоритма на JAVA.*

Ключевые слова: *многоядерные системы ,параллельный генетический алгоритм ,параллелизм ,фреймверк Java concurrent.*

Постановка проблемы. В последнее время активно развиваются многоядерные вычислительные машины, обладающие возможностью многопоточных вычислений. При использовании многоядерных систем необходимо эффективно использовать механизмы распараллеливания процессов для наиболее продуктивного использования вычислительных мощностей, в связи с этим использование параллельного генетического алгоритма видится более продуктивным, чем использование классического генетического алгоритма.

Анализ литературы. Для решения задачи реализации параллельного генетического алгоритма[2] была выбрана миграционная стратегия[1].Для реализации данной стратегии был выбран фреймверк Java Concurrent[7] ,так как содержит множество механизмов для распараллеливания, которые эффективно решают множество проблем возникающих в ходе решения оптимизационной задачи[4].

Цель работы состоит в том, чтобы практически реализовать концепцию параллельного генетического алгоритма с помощью фреймверка Java concurrent на многоядерных системах.

Параллелизм ГА. Генетическим алгоритмам изначально присущ внутренний параллелизм, и одним из способов распараллеливания ГА является разбиение популяции на несколько отдельных подпопуляций[8] ,каждая из которых будет, независимо от других подпопуляций, обрабатывается ГА[3] .Взаимодействие между ними осуществляется с помощью механизма миграций. Этот тип параллелизма – миграционная

модель – наиболее эффективно реализуется в вычислительных системах путем разделения общей популяции на отдельные подпопуляции, эволюционное моделирование[5] которых осуществляется в отдельных потоках многоядерной системы . Если между отдельными подпопуляциями установить правила миграции отдельных индивидуумов, то получится миграционная модель. Для построения миграционной модели требуется задать количество и размеры подпопуляций, топологию связей между ними, частоту и интервал миграции, а также стратегию эмиграции и иммиграции[6]. Размеры и число подпопуляций определяются числом и быстродействием используемой многоядерной системы, а также сложностью решаемой оптимизационной проблемы.

Построение ПГА с помощью Java Concurrent. Фреймверк Java Concurrent содержит широкий набор средств для распараллеливания и дальнейшей синхронизации параллельных процессов. Одним из таких наборов являются параллельные коллекции. Параллельные коллекции облегчают разработку многопоточных программ, предоставляя потокобезопасные, удачно сделанные структуры данных. Однако в некоторых случаях разработчику нужно сделать еще один шаг вперед и подумать о регулировании и/или ограничении выполнения потоков. Так как пакет `java.util.concurrent` призван упростить многопоточное программирование, именно поэтому в него включены утилиты для синхронизации. Модель миграции представляет популяцию как множество подпопуляций. Каждая подпопуляция обрабатывается отдельным потоком (Thread_N)[10]. Эти подпопуляции развиваются независимо друг от друга в течение одинакового количества поколений ,по истечению времени изоляции (время изоляции будет зависит от параметра – Sleep,который будет вызывается на текущем Thread,то есть на текущей подпопуляции – Thread_N.sleep(время изоляции)) происходит обмен особями между подпопуляциями .Количество особей ,подвергшихся обмену ,метод отбора особей для миграции и схема миграции определяет частоту так называемого генетического многообразия в подпопуляциях и обмен информацией между подпопуляциями. Отбор особей для миграции происходит с помощью пропорционального отбора. Для миграции берутся наиболее пригодные особи. Так при использовании пропорционального отбора в неограниченной миграции сначала формируется коллекция из наиболее пригодных особей ,отобранных по всем подпопуляциям. Так как в каждом потоке Thread_N обрабатывается своя подпопуляция, то и особи каждой подпопуляции содержаться в коллекции своего потока выполнения. Используются коллекции типа HashSet, в которых содержаться искомые подпопуляции. Ниже приведен фрагмент кода на Java ,который показывает представление подпопуляции в виде коллекции типа HashSet :

```
Set <String> subpopulations_1=new HashSet<String>();
Set <String> subpopulations_2=new HashSet<String>();
Set <String> subpopulations_3=new HashSet<String>();
…
Set <String> subpopulations_N=new HashSet<String>();
```

Генеральная ,либо «общая» популяция представлена в том же виде :

```
Set <String> unionpopulations=new HashSet<String>();
```

На основе такого подхода формируется «общая» коллекция ,которая содержит особи из каждой коллекции подпопуляций. Случайным образом из этой «общей» коллекции выбирается особь , и ею заменяют наименее пригодную в подпопуляции N. Аналогичные действия проделываем с остальными подпопуляциями. При таком подходе так же возможно ,что какая-то подпопуляция получит дубликат своей «хорошей» особи. Весь этот механизм обмена особями подпопуляций реализован через так называемый пул потоков. Пул потоков[9] предлагает решение и проблемы издержек жизненного цикла потока, и проблемы пробуксовки ресурсов. При многократном использовании потоков для реализации новой подпопуляции, издержки создания потока распространяются на многие задачи. В качестве бонуса, поскольку поток уже существует, когда прибывает запрос, задержка, произошедшая из-за создания потока, устраняется. Таким образом, запрос может быть обработан немедленно, что делает наше приложение более быстрореагирующим. Ниже представлена схема работы с пулом потоков :

```
public class WorkQueue
{
   private final int nThreads;
   private final PoolWorker[] threads;
   private final HashSet queue;
   public WorkQueue(int nThreads)
   {
      this.nThreads = nThreads;
      queue = new HashSet();
      threads = new PoolWorker[nThreads];
      for (int i=0; i<nThreads; i++) {
         threads[i] = new PoolWorker();
         threads[i].start();
      }
   }
   …
```

В данном фрагменте кода представлена очередь потоков. Каждый поток содержит свои подпопуляции особей ; пул потоков обеспечивает и безопасный обмен особями ,исключая взаимные блокировки потоков, что в свою очередь обеспечивает лучшую производительность и меньшее время на выполнение операции миграции между подпопуляциями.

Выводы. Для выполнения поставленной задачи – реализации параллельного генетического алгоритма была выбрана стратегия миграции особей между подпопуляциями ,так же называемая «миграционной моделью». Для практической реализации был выбран объектно-ориентированный язык Java и фреймверк для распараллеливания Java Concurrent.Для синхронизации параллельных потов используется пул потоков, который обеспечивает безопасный обмен особями ,исключая взаимные блокировки потоков, что в свою очередь обеспечивает лучшую производительность и меньшее время на выполнение операции миграции между подпопуляциями.Было установлено ,что мощность эволюционных алгоритмов усиливается с применением параллельных вычислений. Использование параллельных алгоритмов дает большую выгоду в производительности системы и времени выполнения поставленной задачи.

Список литературы

1.Курейчик В.М., Кныш Д.С. Параллельный генетический алгоритм. Модели и проблемы построения // -С.1-3.

2.Интернет-ресурс. - Режим доступа:www/ URL: http://www.gotai.net/ - сайт по ИИ.

3. Гладков Л.А., Курейчик В.М., Курейчик В.В. Генетические алгоритмы [Текст]/под ред. В.М. Курейчика . – 2-е издн., испр. и доп. – М.:ФИЗМАТЛИТ ,2006 .-320 с.

4. Иванов Д.Е., Чебанов П.А. Взаимодействие компонент в распределённых генетических алгоритмах генерации тестов / Д.Е. Иванов, П.А. Чебанов // Наукові праці Донецького національного технічного університету. Серія: “Обчислювальна техніка та автоматизація”. Випуск 16(147).- Донецьк: ДонНТУ.- 2009.- С.121-127.

5. Grosso P.B. Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model// Unpublished doctoral dissertation, The University of Michigan. (University Microfilms №8520908), 1985.

6. Курейчик В.М., Родзин С.И. Эволюционные алгоритмы: генетическое программирование. Обзор // Известия РАН. Теория и системы управления.- 2002.-№1.-С.127-137

7. SDN for Java Concurrent / Интернет-ресурс. - Режим доступа: http://docs.oracle.com/javase/1.5.0/docs/api/java/util/concurrent

8. Mitchell M. An Introduction to Genetic Algorithms [Текст]/M.

Mitchell. — Cambridge: MIT Press, 1999 —158 с. — ISBN 0-262-13316-4 (HB), 0-262-63185-7 (PB)

9. Java Concurrency in Practice Brian - Goetz, Tim Peierls, Joshua Bloch - Москва: Sun Press, 2012 —983 с. — ISBN 0-262-13316-4 (HB), 63185(P)

10. Java 2. Библиотека профессионала. Том 1. Основы - Кей С. Хорстманн, Гари Корнелл. - Москва: Sun Press, 2012 —989 с. — ISBN 0-262-13316-4 (HB), 0-262-63185-7 (PB)