Методы обеспечения отказоустойчивости для ряда задач в распределенном окружении

А.А. Московский[1], Е.О. Тютляева[2]

ЗАО «РСК СКИФ»[1], ИПС РАН им. А.К. Айламазяна[2]

Увеличение масштабов современных вычислительных систем приводит к увеличению вероятности отказа отдельных элементов системы. В ряде случаев вычислительные алгоритмы, использующие методы типа Монте-Карло, генетические алгоритмы и т.п., допускают потерю части промежуточных результатов вычислений, что может произойти, например, из-за отказа одного узла в большом вычислительном кластере. В статье предлагаются методы для реализации таких алгоритмов и обеспечения их работоспособности при условии программных и аппаратных сбоев на вычислительных узлах.

В рамках данной работы создан отдельный тип для организации отказоустойчивых вычислений — неготовая переменная (англ. future) с заданным временем жизни. Этот тип обладает всеми свойствами неготовой переменной, поддерживает корректную работу в распределенной памяти, но в шаблонных параметрах данного типа можно указать время жизни такой переменной. Добавление концепции времени жизни означает, что после истечения указанного времени результат переменной ожидаться не будет и программа будет корректно завершена, даже если результат вычисления каких-то функций, ожидаемых в данной переменной, не получен. Программист имеет возможность обрабатывать полученный результат согласно логике задачи, т.е. он может оставить данный результат невостребованным, запустить аналогичную или перезапустить эту же задачу с ожиданием результата в другой неготовой переменной со временем жизни, при этом увеличив время жизни и т.п. Данный подход позволяет получить результат за детерминированное время для тех задач, которые позволяют получить корректный результат даже если одна или несколько вычислительных подзадач не завершили свою работу. Достоинством такого подхода является гарантированная завершаемость при отказах, простота применения и минимальные накладные расходы. Недостатком такого подхода является сложность применения в случае неравновесных гранул параллелизма, к примеру при рекурсивном вычислении.

В качестве альтернативы был реализован метод сохранения задач, при котором все исполняющиеся задачи сохраняются в локальных map-структурах участвующих в вычислении узлов. В случае перехвата отказа одного из вычислительных узлов, все оставшиеся активными узлы удаляют адрес данного узла из списка доступных для проведения вычисления и выполняют повторную посылку тех задач, которые были на него отправлены. Недостатком такого подхода является наличие высоких накладных расходов на сохранение задач и требуемой оперативной памяти на поддержание списка задач. Достоинством является движение в сторону локальной синхронизации, т.к. каждый узел помнит те задачи, для которых было проведено локальное назначение.

Для испытания разработанных примитивов была модифицирована библиотека шаблонных классов C++ T-Sim, в которую была добавлена корректная обработка сбоев и модифицирован планировщик для работы в нестабильном окружении. С использованием разработанных примитивов на языке T-Sim был реализован пример редукционного (монотонного) объекта, а также нескольких альтернативных механизмов перезапуска заданий.

Чтобы гарантированно выявлять сбои, а не ждать завершения работы задачи, направленной на аварийный узел, в тех случаях, когда больше никакие задачи на этот узел не направляются и таким образом невозможно выявить сбой в работе узла во время естественного хода вычисления, планируется также реализовать дополнительную стратегию планирования, которая будет через заданные промежутки времени опрашивать состояние всех участвующих в вычислении узлов.