

Распределенные алгоритмы и системы

mk.cs.msu.ru → Лекционные курсы → Распределенные алгоритмы и системы

Блок 41

Отказоустойчивые алгоритмы
Модели неисправностей
Задачи принятия решения

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

Отказоустойчивые алгоритмы

Компьютеры с развитием технологий становятся всё более надёжными, и вероятность ошибки в каждом конкретном вычислительном устройстве становится всё ниже

Но при этом с увеличением числа компонентов распределённой системы растёт и вероятность отклонений в работе этих компонентов — например, из-за:

- ▶ выхода компонента из строя
- ▶ помех в канале связи
- ▶ воздействия злоумышленников на узел

Чтобы избежать перезапуска всей системы из-за отклонения в работе отдельных её компонентов, следует устроить алгоритм, выполняющийся в этой системе, так, чтобы он справлялся с возникающими неполадками, то есть обеспечить его **отказоустойчивость**

Отклонение в поведении компонента системы от того, которое предполагается алгоритмом, принято называть **отказом** компонента

Отказавшие компоненты системы также принято называть **неисправными**, а остальные — **исправными**

Отказоустойчивые алгоритмы

Два основных вида отказоустойчивых алгоритмов:

1. **Стабилизирующийся**: при отказе и соответствующем переходе вычисления в «неправильную» конфигурацию после восстановления узла рано или поздно достигается «правильная» конфигурация
 - ▶ Такие алгоритмы предназначены для устранения **кратковременных неисправностей**, как, например, вспышки излучения в космосе
2. **Робастный**: каждый узел выполняет действия настолько «осторожно», чтобы не принимать неправильные решения и не выполнять неправильные действия даже в том случае, если от неисправных узлов приходят ошибочные данные
 - ▶ Такие алгоритмы предназначены для защиты от **устойчивых неисправностей** (как, например, захват узла злоумышленником или поломка части микросхемы) и в тех случаях, когда даже кратковременное нарушение правильности работы системы недопустимо

Далее обсуждаются в основном только робастные алгоритмы

Модели неисправностей

При обсуждении того, для каких видов неисправностей предназначен отказоустойчивый алгоритм, будем использовать следующие допущения:

1. Каналы всегда надёжны и исправны, и неисправными могут быть только узлы
 - ▶ В частности, сообщение, отправленное исправным узлом, рано или поздно доставляется адресату
 - ▶ Отклонения в работе канала можно промоделировать соответствующей неисправностью узлов, инцидентных этому каналу
2. Все пары узлов соединены каналами связи (топология «клика»)

Модели неисправностей

Основные виды отказов в узлах:

1. Изначальное бездействие

Узел не выполняет ни одного действия в вычислении

2. Выход из строя

Узел до некоторого момента работает исправно, а затем не выполняет более ни одного действия

3. Византийский отказ (предательское поведение)

Узел выполняет сколь угодно много каких угодно действий

В частности, он может посылать сообщения с любым содержанием, в том числе пытаясь «сорвать» работу алгоритма

Узел, в котором в начале работы системы происходит византийский отказ, принято называть **византийским**

В робастном алгоритме зачастую некоторые узлы полагаются изначально отказавшими и не восстанавливающимися, и требования корректности формулируются для исправных узлов и их состояний

Задачи принятия решения

При разработке и анализе робастных алгоритмов особое внимание уделяется **задачам принятия решения**, в которых каждый исправный узел должен рано или поздно **принять решение**, необратимо записав некоторое значение (**решение**) в заданную переменную

К распределённым алгоритмам решения таких задач (**алгоритмам принятия решения**) обычно предъявляются следующие требования:

1. Каждый исправный узел должен рано или поздно принять решение и завершиться
2. Существуют по крайней мере два различных решения и вычисления, в которых принимаются все допустимые решения
3. Существует «разумная» взаимосвязь между решениями, принимаемыми в исправных узлах
 - ▶ Например, можно потребовать, чтобы исправными узлами были приняты одинаковые решения — такая задача принятия решения называется задачей **консенсуса**

Задачи принятия решения

Примеры применения задач принятия решений:

Избрание лидера

В этой задаче требуется, чтобы

- ▶ один узел принял решение стать лидером — необратимо записал решение 1, выполнив команду *leader*,
- ▶ а остальные приняли решение отказаться от лидерства — необратимо записали решение 0, выполнив команду *lost*

Согласованные вычисления

Узел в результате отказа может начать отправлять разным исправным узлам разные (противоречащие) версии одних и тех же данных

Чтобы принять «разумное» решение, исправные узлы должны основывать свои решения на одной версии данных, и для этого достичь консенсуса относительно того, какая версия данных правильная

Задачи принятия решения

Примеры применения задач принятия решений:

Согласованные транзакции в базах данных

Транзакция в распределённой базе данных может требовать изменений в нескольких узлах одновременно

Для обработки транзакции

- ▶ каждый узел должен определить, может ли он её выполнить, и
- ▶ узлы, затрагиваемые транзакцией, должны достичь консенсуса по поводу того, следует ли принять (если все узлы могут выполнить) или отклонить (если хотя бы один узел не может выполнить) эту транзакцию