

Събитийно ориентирани изчисления, eventBased алгоритми и данни

Основни възгледи

Всичко около нас може да се реши чрез софтуерната архитектура на събитийен поток от данни и ние всички реагираме на събитията според промените, за които ние осъзнаваме интерес. Аз се опитам да разбера и открия как да оптимизирам нашите и компютърните реакции с цел да се спести време и работа.

Гаро А. Гарабедян
Независимо изследване.

Работата на човек може да бъде обобщена в три основни периодично повтарящи се, но в общия случай не върху един и същ предмет всеки път, стъпки: събиране на данни, прилагане на съображения/ алгоритъм (взимане на решение), произвеждане на външни данни/ команди. Когато алгоритъмът или данните се променят целият процес на работа трябва да започне от начало (преприлагане на новия алгоритъм и/или преприлагане върху новите данни) с цел да се обработи (реагира на) промяната.

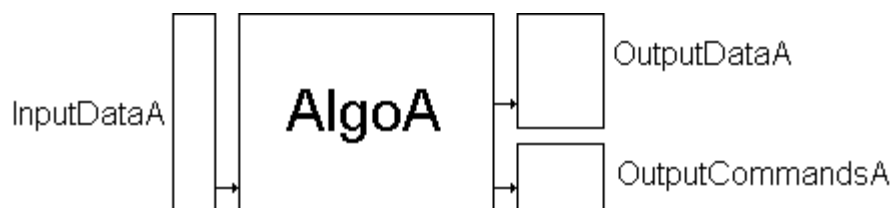
Ако програмата се раздели на сегменти така, че изходните данни на единия да са входни данни на друг(и) то настоящите съображения представят план за изграждане на инструмент за оптимизиране на преизчисленията на входни данни, които са променени от вече обработеното им състояние, която промяна в общия случай задължава наново пълно изчисление.

Преизчисляването е необходимо само, ако произведе различен резултат. Крайният резултат при известно изменение на входните данни е в общия случай единствено предвидим след пълно преизчисляване, а и дори да се развие конкретно за съответния продукт и неговата настояща версия модул, който да изчислява дали ще има някаква промяна на резултата или не, то по настоящем описаната платформа е по-рационална и ефективна при съобразяването на същото.

В сравнение с компютърните програми човек взема също много атомични входни данни, прилага много пъти един алгоритъм и произвежда много изходни данни или команди, или и двете. Когато алгоритъм/ входни данни се променят изпълнителя не преприлага обновения алгоритъм/ входни данни към цялата система, но само към тази част, където променената част от алгоритъма/ входни данни участва в изчисляване (предполага се че ще произведе различни изходни данни в сравнение с настоящото приложение на не обновения алгоритъм/ входни данни). Това поведение

на съобразяване е основата на оптимизирането изложено в този текст.

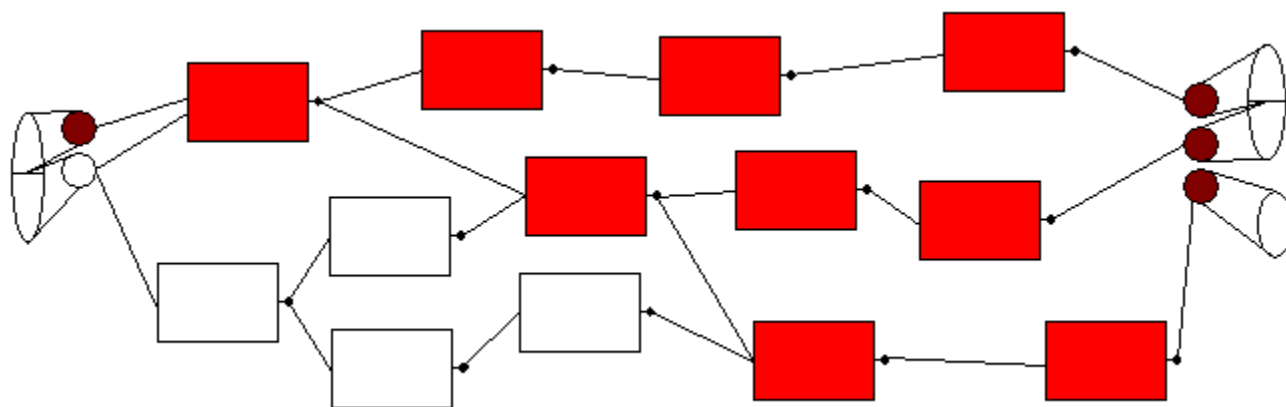
Аз предлагам да се декларира всеки алгоритъм със своите изчислителни атоми и всички данни със своите структурни части (по-малки по обем данни). С цел да се проверява след всяка стъпка (изпълнение на един изчислителен атом) дали съществува разлика между резултата от последното изчисление и промяната (входни данни, алгоритъм или и двете). Атомите на алгоритмите работят само с група от или една входни атомични данни.



Изображение 1: Атомът на работа, входни данни, алгоритъм, изходящи данни.

Бележка: Всичко казано тук и до края на документа за изходни данни и изходни команди носи смисъла на относимост едновременно и към двете.

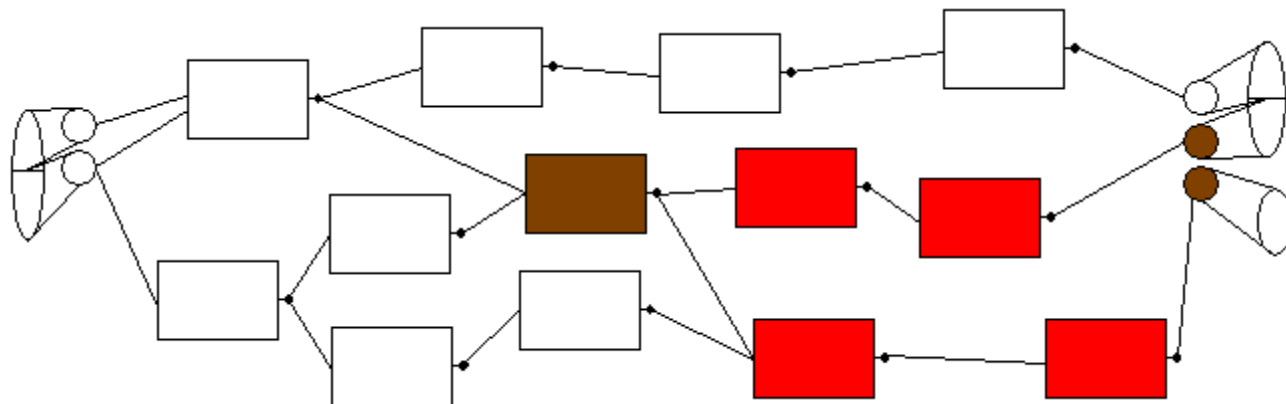
Промяната на входните данни (оцветени в сиво) ще предизвика необходимост от допълнителни преизчисления (маркирани в червено) и възможна промяна на изходните данни (оцветени в кафяво). Стъпките на допълнителни обработки са изпълнени в съгласие с главния алгоритъм:



Изображение 2: Процес на преработка, когато една входни данни е променена.

Промяната на алгоритмичен атом (маркиран в кафяво) ще предизвика необходимост

допълнителни изчисления (оцветени в червено) и възможна промяна на изходните данни (маркирана в кафяво). Стъпките на допълнителна преработка са съобразени с главния алгоритъм. Изходните данни на променения алгоритмичен атом (в кафяво) се преизчисляват първи:



Изображение 3: Процес на работа и как той трябва да се преповтори, когато един алгоритъм е променен.

С оптимизационни съображения, проверяваме след всяка стъпка дали има разлика между последното изчисление и новото (входни данни, алгоритми или и двете).

Съществуват входни данни, които не можем непрекъснато да следим. Не е рационално да се обръщаме към тях непрекъснато, за да проверяваме за възможна промяна. В такъв случай би трябвало да се обръщаме към тези входни данни в най-лошия случай, когато данните му са пряко свързани с належащо изчисление, а не с оптимизационно такова.

Компютърни алгоритми, компютри прилагат алгоритми

В компютърното приложение на алгоритми машината трябва да е в състояние да запази всичките входни и изходни данни. При това положение, приложението трябва да бъде написано на достатъчно абстрактна платформа, която да позволи извличане (и бъдеща инжекция) на цялото количество и стойностите на входни и изходни данни и команди.

От разработническа гледна точка, намирането и декларирането на колкото се може повече атомистични части на алгоритмите с цел да се позволи на технологията с голям процент на ефективност да търси и преизчислява само необходимите входни данни, когато промяна е приложена, е рационално до колкото изчисленията за съобразяване на различие в резултат (сравняване на изходните данни) не станат съществено много като количество и не заемат

съществено изчислително време.

Може да не се декларира, а платформата да пази резултата от всяка операция и всеки път да го сверява, но тази подредба силно ще забави процеса на изпълнение на приложението и няма да е ефективна, ако се правят вътрешни изчисления и в процентно отношение приложението не чака външни данни (операции), а повече изчислява със своя изчислителна мощност.

Редактори на съдържание, потребителя обработва данни с помощта на компютърно приложение

Прилагайки това към човешки труд, ние не можем да разпознаем всички промени на данни и алгоритъмът, който хората прилагат. Но когато някой промени част от документ може да бъде информиран, че съществува декларация за части от този или друг документ, които според първата са относими към току що променените данни.

Данните в човешката работа са свързани по тяхната относимост към проблема, който човек (потребителя) разглежда. Разрешаването на технологичното съхранение на тези връзки може да предпази потребителя от допускане на грешки, когато редактира изолирани части от голям документ. Без запознаването с целия документ и опитвайки се да се намерят отношенията между неговите части и съобразяването на същите при редактирането на някоя част от документа, потребителят може да прегледа всички деклариранни части в документа. Един атом за данни може да съдържа много части от документ, които от своя страна да участват и в други атоми.

Няма теоретична пречка част данни да са относими към много различни части.

Защото хората прилагат различни алгоритми на съдържание и това количество алгоритми не е крайно и защото връзките между частите на документа е функция на приложеният алгоритъм. Потребителят няма да е 100% защитен от промяната на съдържание и в погледа на целия документ да допусне безсмислица.

Докато потребителя редактира, всички относими части съдържание на редактираната част би следвало да са му представени.

Полезно е да се позволи на потребителя да декларира абстрактни алгоритмични атоми и да моделира процеса на работа върху съдържанието, алгоритмичните атоми в техните отношения помежду си и с атомът за данни.