СЪДЪРЖАНИЕ

CONTENTS

Информационни технологии I Information Technologies

СЪДЪРЖАНИЕ

CONTENTS

Bulgaria Communications Chapter

АЛГОРИТЪМ ЗА УПРАВЛЕНИЕ НА МИКРОКОНВЕЙЕРЕН БУФЕР

Димитър С. Тянев, Виолета Т. Божикова, Стефан К. Герганов, Божидар П. Георгиев

Резюме: Разглеждат се проблеми, свързани с хардуерната реализация на изчислителен процес, съдържащ условни преходи. За хардуера се има предвид асинхронна конвейерна организация на микрооперационно ниво. Характерното за него е, че включва както еднотактни така многотактни микроконвейерни звена. В резултат на тези условия, слизащите от конвейера резултати не съответстват по ред на реда на стартираните в конвейера задачи. В статията са представени: синтезираната оригинална логическа структура на конвейерен буфер, както и специфичната за неговото обслужване стратегия, благодарение на които при четене на резултати от буфера се възстановява техният правилен ред. Описан е още програмен модел на структурата на буфера, с помощта на който е изследвано поведението му в различни възможни за него ситуации. Представени са резултати от числените експерименти с програмния модел, въз основа на които могат да бъдат избрани параметрите на буфера.

Ключови думи: Микроконвейер, Условен преход, Възстановяване на реда, Микроконвейерен буфер.

Algorithm for Micropipeline Buffer Control

Dimitar S. Tyanev, Violeta T. Bozhikova, Stefan K. Gerganov, Bozhidar P. Georgiev

Abstract: The paper focuses on the problem of hardware implementation of the computational process containing conditional transitions. Asynchronous pipelines organization of micro-operational level is provided for the hardware. Its characteristic feature is that it includes both one cycle and multi-cycle micropipeline units. Because of these circumstences, the outgoing pipeline results do not correspond in order to the tasks running in the pipeline. The article presents synthesized original logical structure of the micropipeline buffer and a specific to its service strategy through which their correct order is restored when reading results from the buffer. Another programming model of the structure of the buffer is described, by which its behavior was studied in different possible situations. In addition, a programming model of the structure of the buffer was created and its behavior in different possible situations is examined. The results of numerical experiments with the programming model are presented. Based on them recommendations are formulated about the parameters of the buffer and the structure of the pipeline.

Keywords: Micropipeline, Conditional transition, Restore order, Micropipeline buffer.

I. Същност на задачата

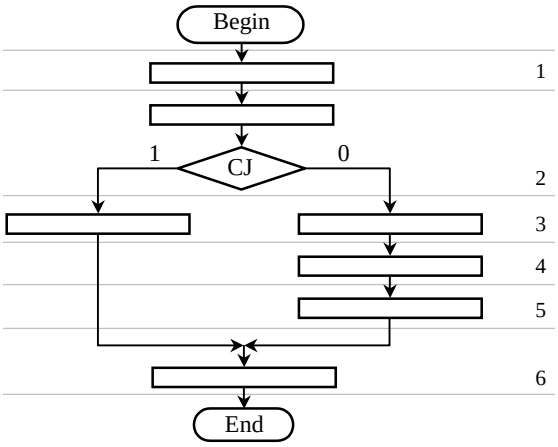
Разглеждаме представената на фигура 1 структура на примерен алгоритъм, който е апаратно реализиран и чието изпълнение е конвейерно организирано. Има се предвид още, че този алгоритъм е детайлизиран и неговите изпълними блокове са реализирани апаратно с помощта на методите, изложени в [1], [2], [3], [4], [5] и [6]. Всеки изпълним блок от блок-схемата представлява отделно еднотактово или многотактово микроконвейерно звено, според определенията, изложени в [7], [8] и [9].

Както се вижда, представеният алгоритъм може да се определи като разклонен. Условието за преход CJ (*conditional jump*) формира в примерния алгоритъм следните възможни пътища за изчислителния процес:

1. Begin; 1; 2; (CJ=true); 3; 6; End .

2. Begin; 1; 2; (CJ= false); 3; 4; 5; 6 ; End ;

където с 1,2,3,... са означени номерата на нивата, през които преминава изпълнението.



Фиг. 1 Структура на примерен алгоритъм

Като се има предвид, че всеки алгоритмичен път *Begin-End* е уникален, съответните последователни микроконвейерни звена в паралелните клонове са отнесени към едно и също поредно ниво на конвейера, за който тук те са общо 6. Така, към ниво 3 например, са отнесени две микроконвейерни звена. В момента на изпълнение на всяка отделна задача, достигнала това ниво, съобразно стойността на условието за преход, работи само едно от звената (или лявото или дясното). В общата точка (входа на 6-то ниво), където се обединяват двете разклонения, заявките, придружаващи получените в тези клонове междинни резултати, задължително се отнасят до различни задачи, стартирани преди това в микроконвейера. Редът, в който тези резултати достигат точката на обединяване, съвсем не може да се очаква да съответства напълно на реда, в който са стартирани задачите, на които те принадлежат. С други думи, в приемащото звено, което стои в общата точка, едва ли ще постъпват данни в правилния ред. Основна причина за тази състезателност е самото разклонение. Като се има предвид различния брой микроконвейерни звена в клоновете, както и внасяните от тях закъснения, съвсем възможна става ситуацията, при която предният фронт на изчислителния процес на по-късно стартирана задача изпреварва този на по-рано стартирана задача, достигайки общата точка. В резултат на тази възможност, не е правилно да се очаква редът на излизащите от конвейера резултати да съответстват напълно на реда на стартираните задачи. С други думи, резултатът, излязъл първи, едва ли е резултат на първата стартирана в конвейера задача. Ако за по-нататъшните изчисления редът на резултатите е от значение, то той следва да бъде възстановен. Така изложените съждения представят същността на проблема, на който е посветено това изследване. Този проблем е един от изявените в [10].

Проблемът с възстановяване на реда на слизащи от изчислителен конвейер резултати принципно е разглеждан например в [11], [12]. Описаното там решение е свързано с конвейера на машинните команди в цифровите процесори. В този вид конвейери той се поражда по причина на изкуствено въведения в нивото *execution* паралелизъм, водещ до суперскаларност. В този смисъл, разглежданите от нас тук причини за възникналия на микрооперационно ниво проблем, са коренно различни. И докато на високо (командно) ниво проблемът в голяма степен се решава с програмни средства, то на ниско (микрооперационно) ниво той следва да бъде решен хардуерно.

II. Конвейерна сянка

След направения по-горе извод следва, че резултатите, които се явяват на изхода на конвейера, като резултати на микроконвейерното звено в последното 6-то ниво на примера, вероятно няма да са подредени. Това означава, че тези резултати няма да могат да се

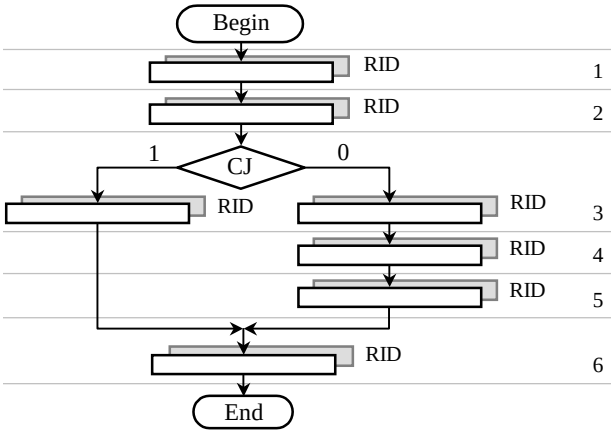
използват веднага за следващи задачи и ще трябва временно да бъдат съхранявани, докато дойде редът им за това. Тъй като от изхода на конвейера непрекъснато ще слизат готови резултати, най-естественото им съхраняване може да се реализира с буферна памет от тип FIFO. За класически конвейери, в които липсват разклонения, типът FIFO на буфера е естествен. В нашият случай, структурата и стратегията за обслужване на буфера е значително по-сложна, но ние ще запазим това му условно наименование. Нека за начало приемем, че FIFO-буферът има толкова клетки, колкото са нивата в конвейера, т.е. съобразно примера те ще бъдат 6. Нека приемем още, че са завършили изчисленията на 6 последователни стартирания на конвейера и принадлежността на резултатите се изразява с последователността: 4,2,1,5,3,6. Така разбираме, че първият резултат, слязъл от конвейера, принадлежи на задачата, която е стартирана четвърта по ред, а резултатът на стартираната като първа е излязъл трети по ред. Последният, 6-ти резултат, е излязъл без да пререди или да бъде изпреварен. Числата в примерната последователност не представляват самите резултати, а само техния стартов номер, т.е. на изхода стартовите номера не са подредени в естествен ред, като с това илюстрират изложеното предположение. Ако след поредния запис буферът се запълни, записването в него трябва да бъде блокирано.

Задачата за възстановяване на правилния ред не може да бъде решена само чрез самите резултати. Необходима е допълнителна информация. Всеки резултат (и междинен и окончателен) трябва да се придружава от идентификатор, който във всеки един момент ще показва принадлежността му към изпълняваната в конвейера задача. Този идентификатор трябва да слиза от конвейера заедно с окончателния резултат на съответната задача. Идентификаторът следва да бъде използван в последния етап, когато резултатът трябва да бъде записан в конвейерния буфер на правилното място. От тук следва, че конвейерният буфер ще трябва да се управлява по специален алгоритъм, различен от познатия FIFO-алгоритъм.

За да бъдат материализирани горните съждения ще бъде необходимо допълнително хардуерно оборудване в конвейера. Всяко микроконвейерно звено заедно с входните данни ще трябва да приема и идентификатора на тези данни. Звеното ще трябва да съхранява този идентификатор до края на изчисленията, които провежда и да го предава към следващото звено заедно с резултата, който е получило. Следователно конвейерните регистри следва да бъдат допълнени с регистри за съпътстващите идентификатори. Поради пасивната роля на тези регистри си позволяваме да наречем тяхната съвкупност “сянка” на конвейера, за което фигура 2 ни дава ясна представа.

Предлаганата тук идентификационна схема за възстановяване на реда, който следва да заемат резултатите, не се основава на порядковите номера. Използването на порядкови номера или на адреси [11], [12] във всеки вариант потенциално застрашава алгоритъма за поддържане на конвейерния буфер от фиксиране на едни и същи тагове. Това се дължи на крайната дължина на логическите възли, които могат да ги генерират. Ето защо тук порядковите номера ще използваме само за пояснение на предлаганата нова стратегия.

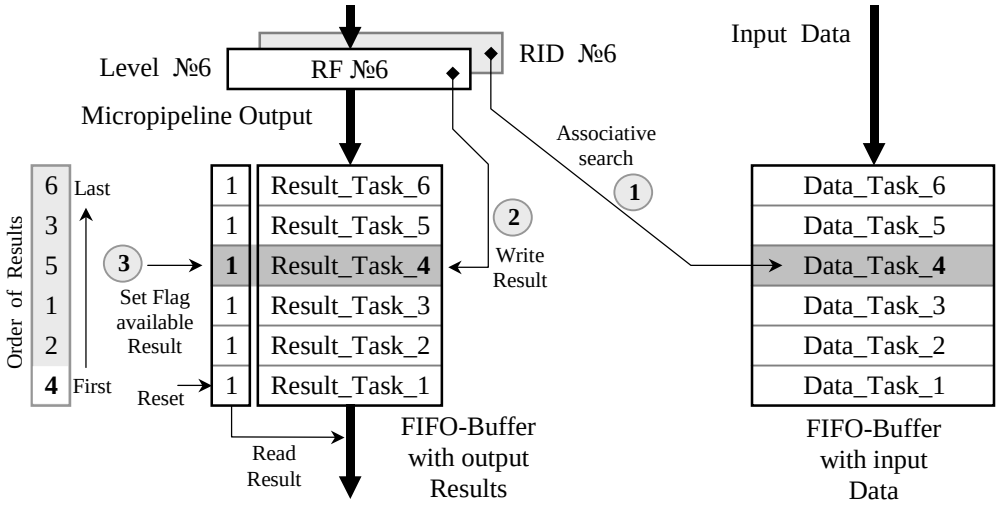
Във всеки един момент и във всяко едно място в конвейера, за идентификация както на междинните така и на крайните резултати, тук се предлага да се използват входните данни на всяка отделна задача. Така входните данни ще придружават получаваните резултати и ще следват техния път от звено към звено по хода на предния фронт на изчислителния процес. В последното звено, непроменени, те ще излязат заедно с окончателния резултат, за който са били използвани. Взетото решение е илюстрирано на фигура 2, където зад микроконвейерните звена се виждат регистрите RID, съдържащи идентификационните данни.



Фиг. 2 Положение на регистрите за идентификаторите – "сянка" на конвейера

III. Конвейерен буфер. Алгоритъм за обслужване

Изходните данни (резултати и идентификатори за принадлежността им) са използвани за решаване на поставената задача по начина, илюстриран с фигура 3:



Фиг. 3 Структура на микроконвейерния буфер

В структурата е показан регистъра-фиксатор RF само на последното микроконвейерно звено, от което "слизат" готовите резултати и зад него – регистъра с идентификационните данни RID, придружаващи резултата. Структурата на микроконвейерния буфер съдържа FIFO-буфер, в който последователно се записват входните данни на всяка новостартирана в конвейера задача. Обърнете внимание – входните данни се подреждат в този буфер в правилния стартов ред. Като пример за работата на конвейера е посочена последователност от 6 задачи, завършването на които се предполага във вече посочения ред – 4,2,1,5,3,6.

Паралелно към буфера на входните данни е поставен FIFO-буфер за изходните резултати. Всяка клетка на този буфер притежава допълнителен бит, представляващ признак (флаг) за съществуване (за присъствие) на резултат, който още не е изчетен, т.е. все още се съхранява в буфера.

Началното състояние на микроконвейерния буфер следва да бъде определено принудително, при което всички негови полета са нулирани.

Предлаганата буферна структура обслужва микроконвейера по следната оригинална стратегия:

III.а Запис в микроконвейерния буфер

1. В момента, в който на изхода на конвейера се получи готов резултат, придружаващите го идентификационни данни, използвани като асоциативен признак, извършват асоциативно търсене в буфера с входните данни. На фигурата е показан случай с резултата на първата готова задача №4. Съвпадението (RID№6)≡(Address 4) осигурява достъпа до същия адрес (същата линия) в буфера на изходните резултати. В този смисъл микроконвейерният буфер работи като асоциативна памет ;
2. В така достъпната клетка на буфера за изходни резултати се записва готовият резултат. Според примера, първият излязъл резултат е на задача 4. Заедно с този запис се извършва още и;
3. Запис на единица в присъединения към отворената клетка бит за присъствие.

Процесът на запис на слизащите от конвейера резултати продължава по този алгоритъм до момента, в който се получи резултат, чийто порядков номер изисква запис в изходната (най-долната) клетка на FIFO-буфера. Според логиката на FIFO-буфера с входните данни, управляващ буфера с резултатите, така записаният резултат принадлежи на задача, чийто порядков номер точно съответства на входящия. От всички задачи, работещи в конвейера, тази, чийто резултат следва да бъде записан в изходната клетка на буфера за резултати, ще наричаме най-ранната задача. С други думи записаният в тази клетка резултат подлежи на незабавно четене. Тази ситуация може да се разпознае лесно по установената единица в бита за присъствие на тази изходна клетка. С прочитане на резултат от изходната клетка на FIFO-буфера, същата се счита за свободна, след което се реализира всеобщо придвижване на данните към изход.

III.б Четене от микроконвейерния буфер

Преди да поясним операция четене, трябва да подчертаем още веднъж, че в резултат на поведението при операция запис, резултатите на отделните задачи са вече подредени в буфера на изходните резултати в пълно съответствие с реда на стартирането им. Така тяхното четене може да бъде последователно и напълно съответстващо на дисциплината FIFO.

Четенето на резултатите, получени от микроконвейера, е задача на външно устройство, което тук не е обект на внимание. Все пак следва да бъде отбелязано, че на четене подлежи само изходната клетка на FIFO-буфера. Съдържанието на тази клетка е достъпно за четене само когато в нейния бит за присъствие има записана единица. Тази единица играе ролята на сигнал за разрешаване на операция четене, както е илюстрирано на фигура 3.

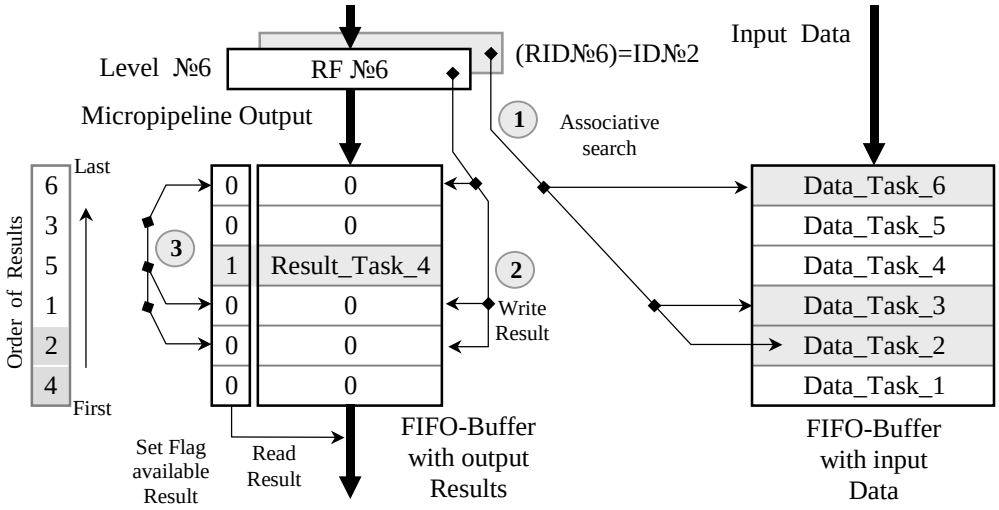
След прочитане на резултата от изходната клетка на буфера, цялата линия от полета може да се приеме за свободна. Според дисциплината за обслужване FIFO, съдържанието на буфера се премества към изхода, при което се освобождава входната му линия, в която може да бъде реализиран следващ запис.

III.в Особени ситуации в алгоритъма

Изказаният по-горе алгоритъм следва да се прецизира допълнително. Причина за това са различни ситуации, които могат да бъдат създадени както от входните данни, така и от параметрите на изчислителния процес в конвейера. Имаме предвид както параметрите на конструкцията на отделните микроконвейерни звена, така и на закъсненията, които допълнително зависят още и от конкретните данни.

IV.г Задачи с еднакви данни

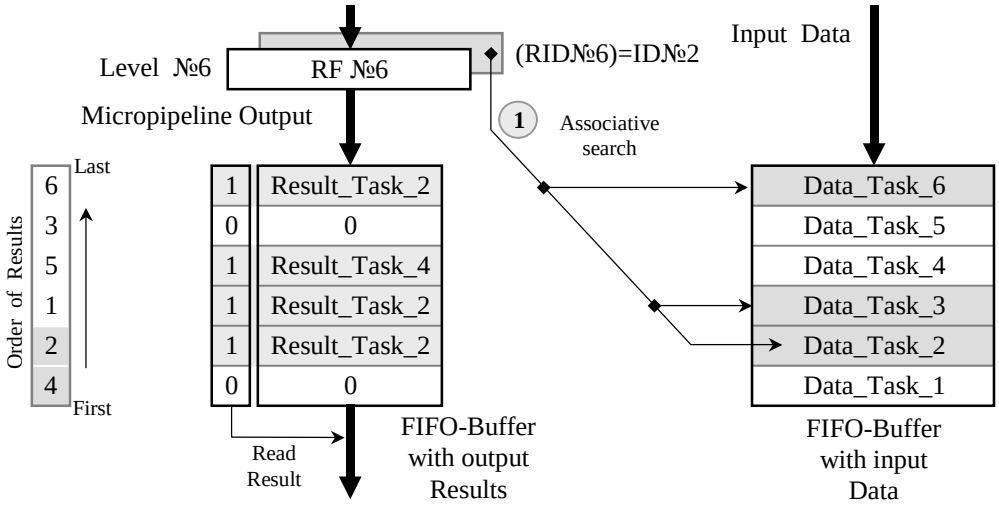
За начало ще се спрем на ситуация, при която на входа на конвейера, поредно зарежданата задача има входни данни, които съвпадат с входните данни на преди това стартирани задачи, все още съдържащи се в буфера. Тук трябва да отбележим, че ако една задача има входни данни, които съвпадат с тези на преди това стартирала задача, то нейният алгоритмичен път *Begin-End* ще бъде същият. От това следва, че по-късно стартиралата задача не може да изпревари изпълнението на преди това стартиралата задача. Освен това резултатът й ще бъде същият. При описаното по-горе поведение, фактът, че входните данни съвпадат с тези на други задачи, ще бъде установен в момента, когато първата от така стартираните с еднакви данни задача слиза от конвейера. В този момент (пункт №1 в горната последователност) асоциативното търсене в буфера на входните данни ще установи съвпадение с вече съществуващи записи. Тази ситуация е илюстрирана на фигура 4, където се предполага примерно, че задачи 2, 3 и 6 в предполагаемата последователност 4,2,1,5,3,6, имат едни и същи входни данни.



Фиг. 4 Случай на 3-кратно асоциативно съвпадение

На фигура 4 е изобразено, че са стартирани 6 задачи, и че първата завършила задача 4 е записала своя резултат. При завършване на следващата задача 2 (както е показано на фигурата) се установява асоциативно съвпадение с признака на задача 3 и на задача 6 (вижте оцветените в жълто клетки). Така възниква въпросът: в коя клетка да се запише слизащият от конвейера резултат на задача 2? Възможностите са две: да се запише само в своята си клетка, както това направи задача 4, или да се запише във всички клетки, генерирали асоциативното съвпадение. Тъй като при еднакви входни данни всички тези задачи (2, 3 и 6) ще получат еднакви резултати, вторият вариант за запис само изпреварва във времето нормалния запис на резултатите за задачи 3 и 6. Това означава за примера, че изчисленията, които ще провеждат задачи 3 и 6 (като задачи следващи вече завършилата №2) ще продължават да се изпълняват в конвейера, но на изхода му могат и да не записват своите резултати, тъй като съответните им клетки ще имат вече вдигнат в единица признак за присъствие. При това положение изчитането на данни ще бъде в правилния ред. Ако обаче се приеме да не се усложнява алгоритъма за управление на буферната памет, задача 3, както и задача 6, при завършване могат да записват своите резултати, без при това да се влияят от вече съществуващия в тях запис.

Ако приемем първия вариант на поведение за така извършваните записи с резултата на задача 2, временното съдържание на микроконвейерният буфер ще бъде каквото е показано на фигура 5.



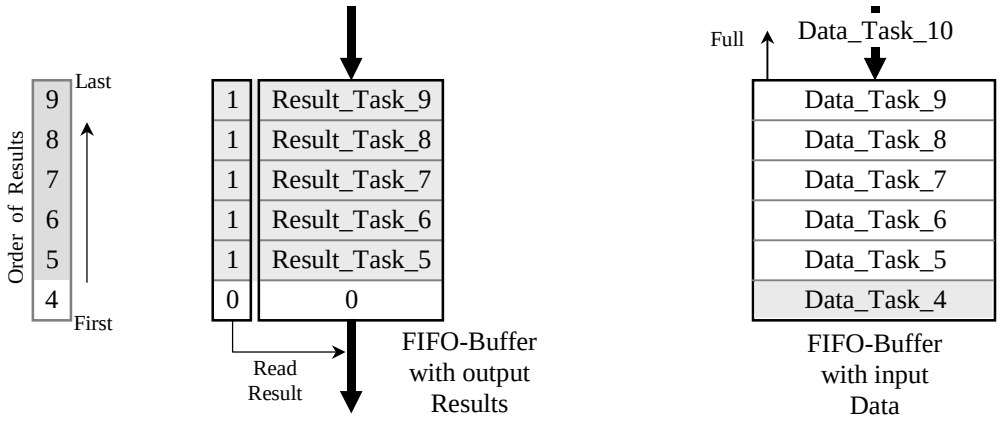
Фиг. 5 3-кратно дублиране на резултата от задача №2

III.д Запушване на алгоритмични пътища

На второ място ще се спрем на ситуация, при която поредната задача не може да бъде заредена в конвейера, защото не може да се осъществи запис на нейните входни данни в буфера. Запис не може да се извърши когато буферът за входни данни е запълнен. Ще разгледаме примерна ситуация върху разглежданата структура от фигура 1, при условие, че обемът на микроконвейерния буфер продължава да бъде 6 клетки.

Приемаме, че първоначално в конвейера са стартирани последователно 6 задачи. Така входните данни на тези задачи са запълнили буфера за входни данни. Ще приемем още, че изпълнението на задачи 1, 2 и 3 е преминало през клона "лъжа" на алгоритъма (микроконвейерни звена в нива 3, 4 и 5), а изпълнението на задача 4 се е отклонило в клон "истина", където изчисленията в микроконвейерното звено на този клон (ниво 3) са се задържали продължително време. При това условие задачи 1, 2 и 3 достигат ниво 6, завършват успешно и записват резултатите си. Нека приемем, че докато задача 4 продължава да се задържа в левия клон на ниво 3, следващите задачи 5 и 6, поемат по пътя на първите три. Ако задача 4 продължава да се задържа, задачи 5 и 6 ще завършат успешно и ще запишат своите резултати. Когато това стане, в буфера на резултатите ще има само една празна клетка – тази за задача 4. Тъй като резултатите на задачи 1, 2 и 3 ще бъдат маркирани с бита за присъствие, те ще бъдат и безпрепятствано прочетени. В резултат на тези четения в изходната клетка на буфера за резултати ще пристигне празната клетка за резултата на задача 4. Така четенията ще бъдат прекратени. Въпреки, че завършилите задачи са 5 на брой (№1, 2, 3, 5 и 6), само първите три от тях (1,2,3) ще могат да освободят резултата си. По време на изчитане на резултатите на първите три задачи, FIFO-буферът за входни данни ще освобождава входни клетки, което ще бъде предпоставка за зареждане в конвейера на нови три задачи с поредни номера 7, 8 и 9. Ако предположим, че изпълнението на тези задачи тръгне също по десния клон на алгоритъма и изпълнението им изпревари задача 4, те могат да завършат успешно и да запишат своите резултати. Състоянието на конвейерния буфер в този момент можем да илюстрираме с рисунката от фигура 6:

Следващата задача №10 няма да може да бъде заредена в конвейера, защото неговият буфер за входни данни е пълен. Този факт може да бъде оповестен чрез установяване в единица сигнала *Full*. В този момент всички микроконвейерни звена в конвейера са в състояние на очакване, с изключение на звеното, в което продължават изчисления по задача 4.



Фиг. 6 В очакване на резултата на най-ранната задача

В тази ситуация задача 10 не може да бъде заредена поради това, че буферът за входни данни е пълнен *Full*=1. Ситуацията в конвейера удивително прилича на запушването на кръвоносен съд от така наречения тромб.

Когато задача 4 излезе от ниво 3 и премине в ниво 6 тя ще завърши и ще запише своя резултат в буфера на конвейера. От този момент нататък всички фиксирани резултати за задачи от 4 до 9 могат да бъдат четени. В момента, в който в конвейерния буфер се освободи входната клетка, в конвейера ще стартира задача 10, а по-късно и следващите след нея.

Трябва да приемем, че подобно продължително задържане на изчисленията, може да настъпи във всяко многотактово микроконвейерно звено. При това в такова продължително задържане могат да изпаднат няколко звена едновременно, което означава наличие на "тромб" в съответните алгоритмични пътища *Begin-End*. От направения по-горе анализ следва, че при такива ситуации конвейерът не губи своята работоспособност, но значително снижава производителността си. Ако едно микроконвейерно звено често изпада в продължително задържане, то изводът е, че мястото на това звено се изявява като "тясно". Разбира се в такива случаи конструкторът следва да предприеме спрямо конвейера необходимите структурни промени. Това, което е ясно и в този момент е, че обемът на конвейерния буфер има значение за производителността на конвейера.

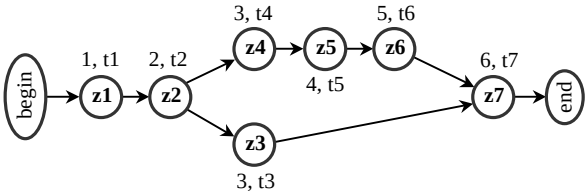
IV. Моделиране и експериментиране на буфера

Върху примерния конвейер и проектирания за него буфер за възстановяване на реда на получаваните резултати беше създаден програмен модел, чрез който системата беше експериментирана и изследвана. В модела конвейерът беше формално представен чрез крайния ориентиран граф G=(Z,U) – фигура 7.

Означенията в рисунката на графа са следните:

- Множеството от върхове Z на графа, с мощност M=|Z|, съответства на множеството микроконвейерни звена (за примера M=7) ;
- Множеството от дъги на графа $U \subseteq X \times X$ представя множеството от връзки между микроконвейерните звена ;
- Чрез множеството N се представя множество от целочислени тегловни коефициенти, съпоставени на върховете Z на графа G, които указват номера на нивото на конвейера, за който се отнася съответния връх z_i, $z_i \in Z$, $i = \overline{1,M}$. Тъй като няколко микроконвейерни звена могат да принадлежат на едно и също ниво, то броят на върховете, които имат еднакви тегловни коефициенти, е ≥ 1 ;

- С буква T е означено множество от тегловни коефициенти t_j, $t_j \in T$, $j = \overline{1,M}$, съпоставени на върховете на графа, изразяващи степента на закъснение, което внасят отделните звена в изчислителния процес.



Фиг. 7 Граф на конвейера G=(Z,U)

При тази формализация възможните пътища *Begin-End* за движение на задачите в конвейера ще бъдат решение на задачата за намиране на всички пътища в графа G=(Z,U), от връх "*Begin*" до връх "*End*", които са два фиктивно върха, включени в него:

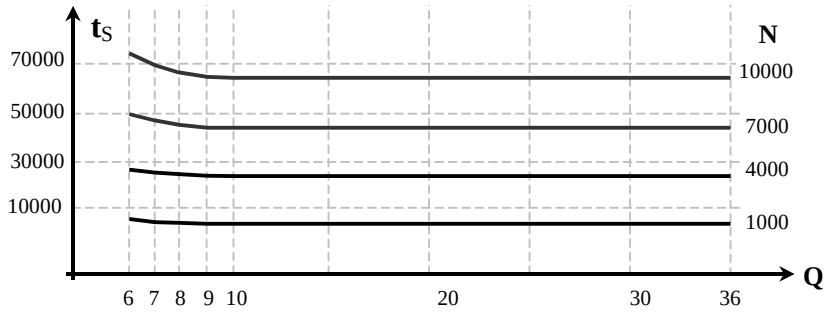
П1: begin–z1-z2-z3-z7-end ;
П2: begin–z1-z2-z4-z5-z6-z7-end .

Предвид спецификата на конкретния случай, при който е налице разнообразие от обработки в конвейера и с оглед на най-висока ефективност по отношение на памет, графът G=(Z,U) е програмно представен чрез динамични списъци на съседство. При това всеки възел се описва чрез структура, която съдържа полетата: входните данни (структура), забавяне за операцията в звеното и връзки със съседни възли (указатели).

Входният и изходният буфери са реализиран като масиви от структури. Във входния буфер, структурата описва входните данни, а в изходния – крайните резултати и готовността им за четене. Алгоритмите, боравещи със структурите, реализират описания алгоритъм на работа на конвейерния буфер.

Програмният модел имаше за цел експериментиране на съвместното функциониране на конвейера с буфера, алгоритъма за обслужване на последния, както и влиянието на неговите параметри върху производителността на системата като цяло. За всеки възел в програмата беше избрана подходяща операция. Входните данни за всяка стартирана задача бяха генерирани случайно. Закъснението при изчисляване на резултата във всяко отделно звено се отчиташе в брой програмни такта. Това позволи получаването на количествена оценка на производителността на конвейера при известен обем на конвейерния буфер и при известен брой задачи, заредени за изпълнение.

Основният въпрос, на който трябваше да отговорят проведените експерименти, беше: как се влияе производителността на конвейера от обема на конвейерния буфер. За всеки експеримент обемът на буфера **Q** беше задаван в интервала [6, 36] клетки, а броят **N** на стартираните задачи от 1000 до 10000. Полученото време за изпълнение на зададения брой задачи $\mathbf{t}_S$ е илюстрирано с графиките от фигура 8.



Фиг. 8 Влияние на обема на буфера върху производителността

Изводът, който е направен въз основа на анализа на получените резултати е, че с увеличаване обема на конвейерния буфер производителността на системата расте експоненциално, като достига насищане, когато обема на буфера достигне броя на звената в конвейера. Този положителен ефект се усилва с броя на изпълнените в конвейера задачи.

V. Заключение

Представеното тук решение на една от четирите задачи, изявени и формулирани в [10], допълва комплексното решение на изследваните проблеми, свързани с проектирането на микроконвейери, реализиращи общи алгоритмични структури. Получаването на решения за тези задачи ще позволи да бъдат снети на микрооперационно ниво алгоритми, които към настоящия момент се реализират програмно. Хардуерното им изпълнение, съчетано с конвейерната организация, ще доведе до значително повишаване на производителността на системите, които ще ги приложат.

В резултат на анализа на възможните за конвейерния буфер състояния, установихме необходимостта от сигнала *Full*, който трябва да бъде функционално свързан с конвейерния автомат на началното (стартовото) звено на конвейера. Конвейерният автомат на стартовото звено следва да бъде блокиран, ако е достигнат от сигнала *Full*=1. Това означава, че автоматът на стартовото звено на конвейери, които ще притежават изходен буфер за своите резултати, трябва да бъде проектиран различно. Намираме този резултат за съществен, тъй като той е достатъчно общ и съответства на общата цел, спомената в началото.

Литература

[1]. Tyanev D.S., Josiffov V., Kolev S.I., *Operational structures without controlling automata*, International Workshop on Network and GRID Infrastructures, 27-28 Sept 2007, Bulgarian Academy of Sciences, Sofia, Bulgaria.

[2]. Тянев Д.С., Колев С.И., *Метод за реализация на апаратни самоуправляващи се циклически структури*, Годишник на ТУ-Варна, Юбилеен сборник “45 години ТУ-Варна”, 2007, ISSN 1311-896X, стр. 130-135.

[3]. Тянев Д.С., Колев С.И., Янев Д.В., *Метод за реализация на апаратни самоуправляващи се циклически структури - част II*, Списание “Компютърни науки и технологии”, ТУ-Варна, ISSN 1312-3335, година V, брой №2/2007, стр. 23-30.

[4]. Tyanev D.S., Kolev S.I., Yanev D.V., *Micro-pipeline Section For Condition-Controlled Loop*, International Conference on Computer Systems and Technologies - *CompSysTech’09*, 18-19 June 2009, Ruse, Bulgaria, pp. I.4 (1-5).

[5]. Tyanev D.S., Yanev D.V., Kolev S.I., *Method for realization of self-controlling loop apparatus structures*, Fifth International Scientific Conference *Computer Science’2009*, 5-6 November 2009, Sofia, Bulgaria. Proc. ’2009, ISBN: 978-954-438-853-9, pp. 154-158.

[6]. Tyanev D.S., Kolev S.I., Yanev D.V., *Race Condition free Asynchronous Micro-Pipeline Units*, International Conference on Computer Systems and Technologies - *CompSysTech’10*, 17-18 June 2010, Sofia, Bulgaria.

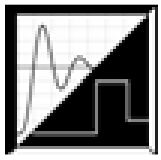
[7]. Тянев Д.С., *4-фазов микроконвейер с еднотактови и многотактови микроконвейерни секции*, Списание “Компютърни науки и технологии”, ТУ-Варна, ISSN 1312-3335, година VII, брой №1/2009, стр. 3-12.

[8]. Tyanev D.S, Popova S.I., *Asynchronous micro-pipeline with multi-stage sections*, ICEST’2010, 23-26 June 2010, Ohrid, Macedonia.

[9]. Kolev S.I., Tyanev D.S., *Early set to zero micropipeline*, International Conference on Computer Systems and Technologies - *CompSysTech’10*, 17-18 June 2010, Sofia, Bulgaria.

[10]. Тянев Д.С., *Управление на разклонения в асинхронни микроконвейери*, Списание "Компютърни науки и технологии", ТУ-Варна, ISSN 1312-3335, година VII, брой №2/2009, стр. 3-12.
[11]. Patterson D.A., Hennessy J.L., *Computer Organization and Design. The Hardware/Software Interface*, 3rd. Ed., Morgan Kaufman Publishers, ISBN 1-55860-604-1, 2005.
[12]. Hennessy J.L., Patterson D.A., *Computer Architecture. A Quantitative Approach*, 3rd. Ed., Morgan Kaufman Publishers, ISBN 1-55860-596-7, 2003.

За контакти:
доц. д-р инж. Димитър Тянев
Катедра "Компютърни науки и технологии"
Технически университет - Варна
e-mail: dstyanev@yahoo.com
д-р инж. Виолета Божикова
Катедра "Компютърни науки и технологии"
Технически университет - Варна
e-mail: vbojikova2000@yahoo.com
Стефан Герганов, Божидар Георгиев
студенти, специалност КСТ
Технически университет - Варна
e-mail: r1g@mail.bg
Рецензент: доц. д-р инж. Н. Рускова, ТУ – Варна



МИКРОКОНТРОЛЕРНО УПРАВЛЕНИЕ НА МОБИЛЕН РОБОТ, СЛЕДЯЩ ЛИНИЯ

Венцислав Ц. Георгиев, Сава И. Иванов, Цоло Т. Георгиев

Резюме: В статията се разглежда начина на организация на обратната връзка на системата за микроконтролерно управление на мобилен робот, следващ линия. Описано е вътрешното периферно устройство таймер/брояч1(T/C1) на микроконтролера ATmega 128, което се използва за реализиране на два широчинно - импулсни модулатори за управление на двете постояннотокови електрозадвижвания, чрез които се управлява движението на мобилния робот. Представен е и алгоритъмът на управление, базиращ се на П закон на регулиране при реализирането на двата затворени контура за управление на постояннотокови електрозадвижвания.

MCU Control of Line Tracing Robot

Ventsislav Ts. Georgiev, Sava I. Ivanov, Tsolo T. Georgiev

Abstract: In this paper is presented how to organize microcontroller control with feedback of line tracing robot. It is described how Timer / Counter 1 unit of MCU Atmega128 works. This unit is used to generate PWM signals for the two DC motors of the line tracer. The main algorithm is presented. It is based on a P controller that controls the two DC motros.

1. Постановка на задачата

Робот следващ линия е този, който следва определен път. Пътят може да бъде видим – черна линия на бял фон (или обратното), а също може да бъде и невидим – магнитно поле.

В тази статия е реализиран вариант на робот, следящ черна линия на бял фон. Управлението става като роботът долавя линията, по която трябва да се движи и маневрира, за да задържи правилния курс на движение. Самото долавяне на линия се реализира с помощта на датчици и се получава една проста, но много ефективна затворена система за управление на робота. На фиг. 1 е показана блоковата схема на реализираната системата.

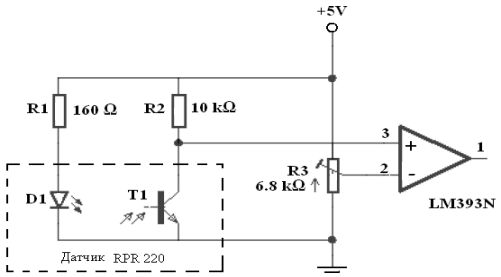
Фиг.1

2. Организация на обратната връзка за управление на мобилния робот

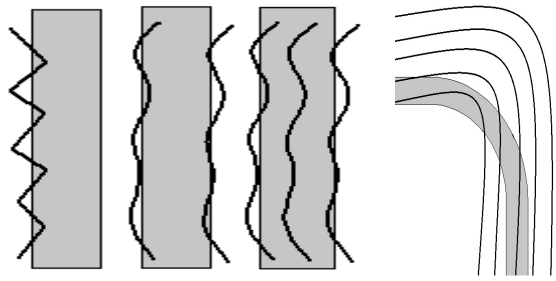
Използваните датчици за следене на линия са фотодатчици RPR 220[4]. Тези датчици са на достъпна цена и лесно се намират в България, което ги прави идеални за подобен тип проекти. Датчикът е съставен от два елемента – светодиод и фототранзистор. Диодът излъчва светлина в инфрачервения спектър с дължина на вълната 940 nm. Транзисторът също работи в инфрачервения спектър и има най – висока чувствителност при дължина на вълната 800 nm. Работата на датчика е много проста. Диодът излъчва светлина и ако тя бъде отразена, част от нея попада върху транзистора. В зависимост от интензитета на светлината съпротивлението на транзистора се променя. При наличието на светлина транзисторът ще има съпротивление клонящо към нула, а при отсъствието на светлина съпротивлението ще

бъде много голямо. Аналоговият сигнал от датчика се подава към компаратора. Целта на компаратора е да формира двоичния вид на данните и да подаде логическа „0” (0 V) за бяло или логическа „1” (+5 V) за черно към микроконтролера. Схемата на свързване за един датчик към компаратор е дадена на фиг. 2.

Съществуват различни подходи за следене на линия [2] и за реализирането на всеки един от тях са необходими някакъв вид датчици, отчитащи линията. След като са избрани датчиците е важно да се уточни техният брой. В настоящата статия са представени някои от най – широко използваните подходи при реализирането на алгоритми за следене на линия.



Фиг.2



Фиг. 3

Управление с един датчик-фиг3а. При управление с един датчик имаме т. нар. „търсене края на линията”. Роботът върви зигзагообразно, като преминава от светло към тъмно. Единият двигател се активира, когато датчикът е върху линията, а когато я изпусне се активира другият. Този вариант е за много ниска скорост и е доста неефективен. Реализацията рядко се прави с помощта на микроконтролер. Датчикът има само 2 състояния – 0 – извън линията, 1 – върху линията.

Управление с два датчика-фиг. 3 b. Това е т. нар. „избягване на линията” В този случай ситуацията е подобна на тази при управление с един датчик, но тук всеки датчик е обвързан с отделен двигател. Целта е линията да бъде заградена през цялото време от двата датчика и роботът да се стреми да я избягва. Този вариант е по – добър от предходния, но при загуба на линията роботът трудно би могъл да отчете това. Вариант за справяне с този проблем е управление с помощта на микроконтролер и реализация на подходящ алгоритъм. Възможните комбинации от състоянието на датчиците са: 00 – колебание между краищата на линията (или загуба на линията), 01 – намерена е дясната страна на линията, 10 – намерена е лявата страна на линията, 11 – не се използва (освен ако датчиците не са разположени на отстояние по – малко то ширината на линията).

Управление с три датчика-фиг.3с. Този вариант на управление е познат като „робот виждащ линията”. Добавяйки трети датчик даваме възможност на робота да разпознава къде е линията и къде са границите ѝ. Това дава предимства при откриване на линията след загуба, по – висока скорост на правите участъци и по – добро справяне със завои. Това е един сравнително добър вариант, позволяващ по – гъвкаво управление с помощта на микроконтролер. Възможните комбинации от състоянието на датчиците са: 001 – отклоняване от линията наляво, 010 – следене на линията, 011 – малко отместване наляво, 100 – отклоняване от линията надясно, 101 – не се използва, 110 – малко отместване надясно, 111 – не се използва (освен ако линията не се пресича или има линия за стоп).

Управление с пет датчика-фиг.3d. Три датчика се оказват напълно достатъчни за доброто управление на робот следващ линия, но за да се постигне по – голяма прецизност при управлението, трябва да се добавят още датчици. Колкото повече датчици се използват в управлението, толкова по – добро става то, което пък открива възможност за увеличаване на скоростта. Не случайно този вариант на управление е добил названието „танцуващ по линията” (фиг. 3d). Възможните комбинации от състоянието на датчиците са: 00000 – линията е загубена заради прескачане на завой или прекъсване на линията (предприемат се

специални действия като търсене на линията и др.), 00001 – линията е почти загубена, рязък завой надясно, 00011 – изпуснат е десния ръб на линията и трябва да завие надясно, 00010 – изместване вляво спрямо центъра на линията, трябва да се извърши корекция надясно, 00110 – леко изместване вляво от центъра на линията, трябва малка корекция надясно, 00100 – движение по центъра на линията, може да се увеличи скоростта и се поддържа права посока на движение, 01100 – леко изместване вдясно от центъра на линията, трябва малка корекция наляво, 01000 – изместване вдясно спрямо центъра на линията, трябва да се извърши корекция наляво, 11000 – изпуснат е левия ръб на линията и трябва да завие наляво, 10000 – линията е почти загубена, рязък завой наляво, 11111 – перпендикулярно пресичане на линията (може да е достигната линия, указваща края на трасето).

Съществуват и варианти за управление с повече от 5 датчика, които предлагат още по – прецизен контрол на движението и възможност за по – висока скорост. Реализираният вариант в тази статия е този с пет датчика

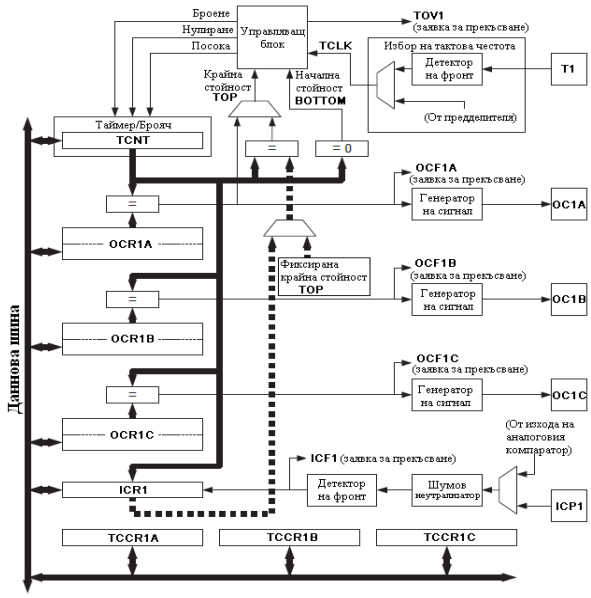
3. Апаратна реализация на канала за управление

След като получи необходимата информация от датчиците, микроконтролерът я обработва с помощта на реализирания за тази цел алгоритъм (описан в т.4). В резултат на това се генерира ШИМ сигнал. Това става с помощта на таймер / брояч 1. Този таймер / брояч е 16 – разреден и в случая служи за генериране на правоъгълни импулси. На фиг. 4 е дадена опростена функционална блокова схема на таймер / брояч 1[1]. Регистърът таймер / брояч (TCNT1), изходните регистри за сравнение (OCR1A/B/C) и регистърът за входно прихващане (ICR1) са 16 – битови. Управляващите регистри на таймера (TCCR1A/B/C) са 8 – битови и няма ограничения за достъп до тях от страна на процесора. Сигналите със заявки за прекъсване са видими в регистъра с флаговете за прекъсване (TIFR) и в разширения регистър с флагове за прекъсване (ETIFR). Всички прекъсвания индивидуално се маскират в регистъра за маски на прекъсванията (TIMSK) и в разширения регистър с маски на прекъсванията (ETIMSK). Регистрите (E)TIFR и (E)TIMSK не са представени в блоковата схема на фиг.4.

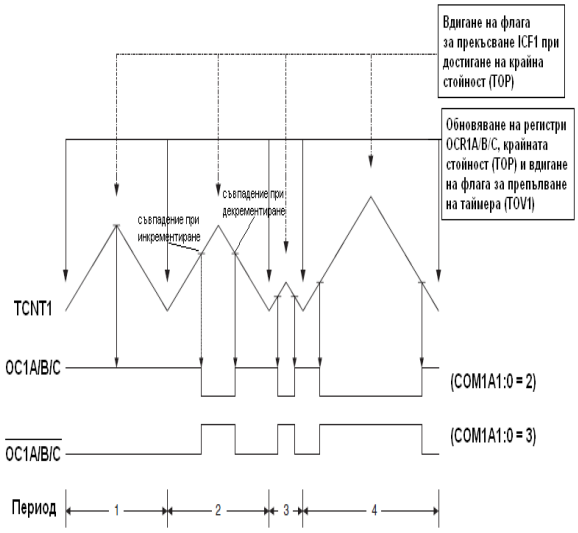
Предназначението на двойно буферираните изходни регистри е да сравняват съдържанието си със стойността в таймер / брояча. Резултатът от сравнението може да бъде използван от блока за генериране на сигнал, за да се генерира ШИМ сигнал или сигнал с променлива честота. Генерираният сигнал се подава на някой от изходите OC1A/B/C. Съвпадението при сравнение ще установи съответстващия флаг за съвпадение (OCF1A/B/C) в единица, което може да бъде използвано за генериране на заявка за прекъсване.

Реализираният при управлението на мобилния робот режим за генериране на ШИМ сигнал е WGM13:0 = 8. Това се дължи на факта, че при управление на двигатели се препоръчва използването на ШИМ с фазова корекция, а още по – добре е използването на ШИМ с фазова и честотна корекция. От двата режима с фазова и честотна корекция е реализиран този с крайна стойност (TOP) ICR1, за да може OCR1A да бъде използван като изходен регистър за генериране на ШИМ. Избраният режим за формиране на изходния сигнал е COM1A1:0 = 2, за управление поведението на пин OC1A, към който е свързан единия двигател и COM1B1:0 = 2, за управление поведението на пин OC1B, към който е свързан другият двигател. При съвпадение при сравнение по време на инкрементиране пинове OC1A и OC1B подават сигнал с ниско ниво (логическа „0”), а при съвпадение при декрементиране се подава сигнал с високо ниво (логическа „1”). На фиг.5 е представена времедиаграма, описваща формирането на ШИМ сигнала

Генерираният ШИМ сигнал се подава към силов преобразовател. Това се налага с цел да се изработи подходящо по форма и големина захранващо напрежение за двигателите. За задвижването на робота са използвани постояннотокови двигатели, чието захранване става с помощта на схема от транзистори (известна като Н – мост) или с готов драйверен чип.

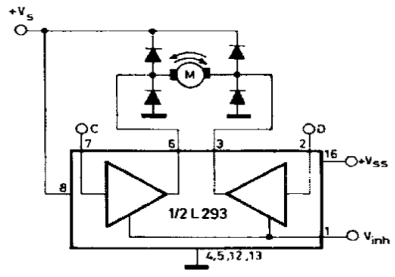


Фиг. 4

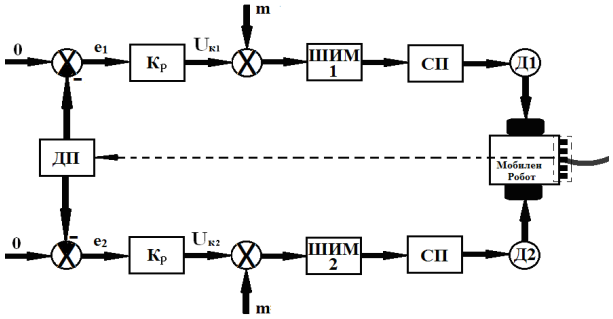


Фиг.5

За целта е избрана интегрална схема – драйвер L293B[3]. Той има 16 пина и 4 канала за управление на един стъпков (по два канала за намотка) или два постояннотокови двигатели. Изходният ток за канал е 1 A (2 A – пиков, моментен), има защита от високочестотни смущения и защита от прегряване. Минималното допустимото захранващо напрежение на чипа е 4.5 V, а максималното – 36 V. На фиг.6 е дадена схемата на свързване за един постояннотоков двигател, с възможност за въртене в две посоки. Към пин 1 се подава генерираният ШИМ сигнал от микроконтролера. Към пинове 2 и 7 се задава посоката на въртене. При подаване на логическа „1” към пин 7 и логическа „0” към пин 2, двигателят се върти в дясна посока. При подаване на логическа „0” към пин 7 и логическа „1” към пин 2, двигателят се върти в посока „напред”. При подаване на еднакъв потенциал към пин 2 и 7, двигателят се спира. Пинове 3 и 6 са изходите, към които се свързва двигателя.



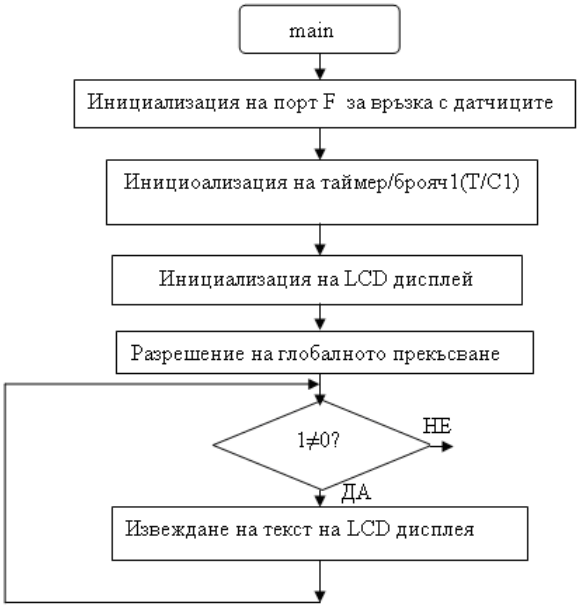
Фиг.6



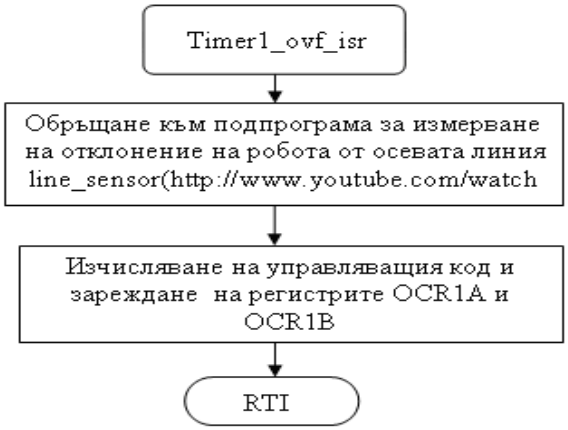
Фиг. 7

4. Алгоритми и програми за управление на робот, следящ линия

Реализираният алгоритъм за управлението на робота се базира на P (пропорционален) регулатор. Управляващия код. получен от програмно реализирания регулатор се дава със следната формула: $u(k) = m + k_p.e(k)$, където

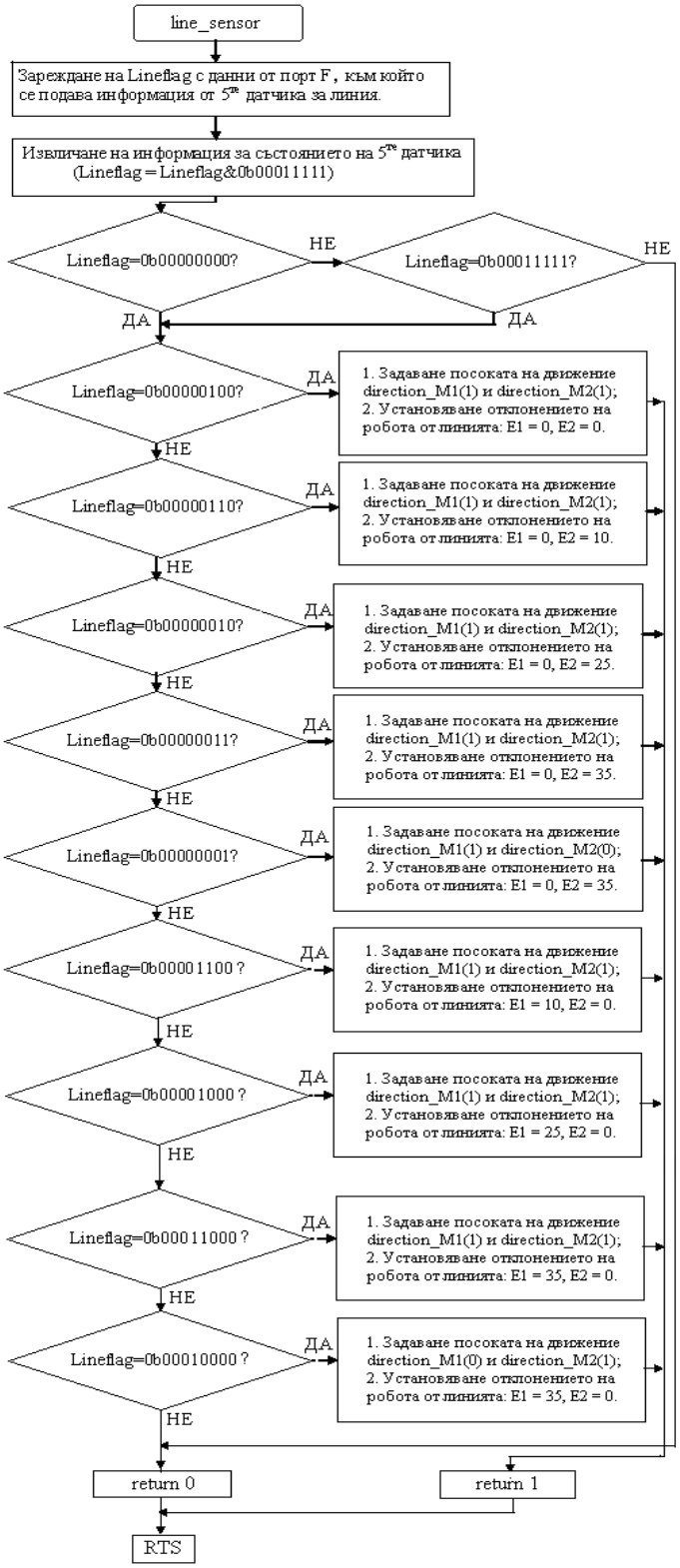


Фиг. 8



Фиг.9

u(k) е изчисленото в текущия такт управ-ляващо въздействие;
m - константа, определяща началното състояние на широчинно импулсния модулатор;
k_p-коефициен на пропорционалност;
e(к)-стойност на грешката на системата в к-тия такт.



Фиг.10

За всеки един от двигателите на робота се организира затворен контур на управление фиг.7. Заданията на двете системи за управление на Д1 и Д2 са равни на нула, което означава, че роботът трябва да се стреми да подържа средния датчик от петте датчика над осевата линия на черната лента(линия).

На базата на експерименти се избират подходящи стойности за K_p=25 (коефициент на регулатора) и за грешките по положение на робота – e_1 и e_2 (с различна тежест в зависимост

от отклонението). С помощта им се генерира управляващ код ($U_{к1}$ = $К_р$.E1 и $U_{к2}$ = $К_р$.E2), който се изважда от зададената стойност m(m=1000). Резултатът е число от 0 до 1023, което се използва за формиране на ШИМ сигнала, който пък се подава към силовия преобразовател (СП), а от там и към двигателите (Д1 и Д2). В зависимост от положението при движение, датчиците за положение (ДП) подават информация, служеща за определяне тежестта на грешките (e_1 и e_2).

Блок - схемите на алгоритмите за управление на мобилния робот са дадени на фиг. 8, 9 и 10. Главната програма за управление на робота се реализира по блок схемата дадена на фиг. 8. С тази програма се решават две основни задачи на управлението:

-инициализация на вътрешните периферни устройства (таймер бтрояч1 и порт F), външното периферно устройство- LCD дисплей и разрешение на глобалното прекъсване;

-осигуряване на фоновия режим на работа на микропроцесора.

На фиг.9 е разгледана блок -схемата на алгоритъма, които се изпълнява при вдигане на флага за препълване на таймер1(TOV1). Чрез този алгоритъм се реализират две задачи: - измерване на отклонението на робота от осевата линия на лентата и изчисляване на управляващите кодове на двата регулатора.

На фиг.10 е дадена блоковата схема на алгоритъма, който обработва информацията, постъпила от датчиците при следене на линия. Използвани са следните означения:

- Lineflag – променлива, в която се записва информацията, постъпила от порт F. Под внимание се взимат младшите 5 бита, отговарящи на петте датчика;

- direction_M1(параметър за посока) и direction_M2(параметър за посока) – функции, задаващи посоката на въртене на двигатели 1 и 2;

- E1 и E2 – тежест на грешката при отклонение от линията.

На базата на този алгоритъм е реализирана подпрограма line_sensor(), която се изпълнява след генериране на заявка за прекъсване от препълване на таймер / брояч 1. Получената информация от изпълнението на подпрограмата се използва от регулатора.

5. Заключение

На базата на алгоритмите дадени на фиг.8, 9 и 10 е написана програма на С за управление на мобилен робота, с помощта на ATmega 128. От направените експерименти се доказва работоспособността на алгоритмите и програмите. Резултатите, получени в статията ще бъдат използвани в учебния процес.

Литература

[1].ATmega128 Datasheet, Atmel Corporation.
[2]. www.wrighthobbies.net
[3] L293B Datasheet.
[4]. RPR-220 Datasheet.

За контакти:
инж. Венцислав Ц. Георгиев
E-mail: venci84@abv.bg
доц. д-р инж. Сава И. Иванов
катедра: Компютърни науки и технологии”
Технически университет - Варна
E-mail: ivanovsi@abv.bg
гл . ас. д-р инж. Цоло Т. Георгиев
кат. ”Автоматизация на производството”
Технически университет - Варна
E-mail: tsologeorgiev@abv.bg
Рецензент: доц. д-р инж. Н. Николов, ТУ – Варна

МОДЕЛ ЗА ПРЕДСКАЗВАНЕ НА ВРЕМЕНА ЗА ИЗПЪЛНЕНИЕ ПРИ ОЦЕНЯВАНЕ НА КЛАС ФИНАНСОВИ ОБЕКТИ*

Ивайло П. Пенев

Резюме: В доклада е предложен модел за предсказване на времената за построение на финансови обекти и изпълнение на симулационни анализи в зависимост от количеството данни, изграждащи обектите. За дефиниране на модела са използвани експериментални данни от реални обекти, съставени от два финансови инструмента с различен брой позиции. Представена е оценка на предложения модел. Целта на модела е използването му за повишаване на ефективността на финансови системи чрез паралелна изчислителна реализация на оценяваните обекти.

A Model for Prediction of Execution Times in the Evaluation of Financial Objects

Ivaylo P. Penev

Abstract: The paper presents a model, predicting the times for building financial objects and executing simulation analyses over the data, forming the objects. The model is defined by the usage of experimental data from real objects, consisting of two financial instruments with various number of positions. An estimation of the proposed model is presented. The target of the model is to be used about increasing the efficiency of financial systems by parallel realization of the evaluated objects.

I. Въведение

Настоящият труд е насочен към повишаване на ефективността на изчислителни системи, изпълняващи оценки на финансови обекти. Оценяването на един обект се състои от построение на обекта и изпълнение на различни симулационни анализи. Тези действия са свързани с прочитане на натрупани статистически данни за обекта от източник (напр. база от данни), извършване на подходящи изчисления с данните и съхраняване на резултати и отчетна информация в същия източник [1]. Симулационните анализи, изпълнявани над отделните обекти са напълно независими помежду си. Възможна е паралелна реализация на задачи, всяка от които оценява даден обект. Ограничително условие за ефективността на такова решение е конкурентният достъп на задачите до общия източник (при четене на данни за построение на обекти и запис на резултати от изпълнените анализи) [1, 2]. Възможен подход за преодоляване на това ограничение е въвеждане на подходящо отместване при стартирането на паралелните задачи [2]. За прилагането на този подход е необходимо с достатъчна точност да са известни времената за изпълнение на оценката над всеки обект.

Всеки обект съдържа различно количество данни. В конкретната предметна област всеки обект е изграден от предварително известен брой финансови инструменти. Според броя и вида на инструментите, както и според вида на изпълняваните симулационни анализи, времето за оценка на обектите се различава. Съществуват предпоставки за формулиране на зависимост на времето за оценка на обект от броя на изграждащите го инструменти при фиксирани симулационни анализи за конкретна изчислителна архитектура.

В представения труд е изследвано оценяването на клас финансови обекти (финансови портфейли), изградени от два типа инструменти в реална изчислителна система. Описани са стъпките за получаване на регресионен модел, предсказващ времето за оценяване на всеки

* Тази разработка е финансово подкрепена от проект: "Подкрепа на творческото развитие на докторанти, пост-докторанти и млади учени в областта на компютърните науки", BG051PO001-3.3.04/13"

обект в зависимост от броя на съставящите го инструменти. Адекватността на модела е проверена чрез статистическо оценяване.

II. Моделиране и оценка на регресионна зависимост

1. Получаване на регресионен модел, формулиращ зависимост на времето за оценяване на обект от броя на съставящите го инструменти.

Изследвани са реални обекти, съставени от два типа финансови инструменти – акаунт и шеър. Обектите се различават по броя на инструментите. За всички обекти се изпълняват еднакви симулационни анализи. Измерени са времената за оценяване на седем различни обекта за конретна изчислителна архитектура.

Таблица 1. Експериментални данни за различни обекти

Брой акаунти	Брой шеъри	Време за оценка (сек)
3	1	32
32	20	118
25	23	91
25	19	110
28	19	113
4	28	110
2	51	231

Целта на моделирането е формулиране на зависимост $y = f(x_1, x_2)$, в която факторните признаци x_1 и x_2 са съответно брой акаунти и шеъри, изграждащи конкретен обект, а y е времето за оценка на обекта в секунди.

Проверява се нелинеен модел:

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_{12} x_1 x_2 + b_{11} x_1^2 + b_{22} x_2^2 \quad (1)$$

За намиране на коефициентите b_0, b_1, b_2, b_{12}, b_{11} и b_{22} се прилага регресионен анализ с използване на метода на най-малките квадрати [3, 4]. По-долу са представени стъпките на класическото решение.

- Намиране на средите на интервалите на изменение - основни нива на факторите.

$$x_{i0} = \frac{x_{imax} + x_{imin}}{2} \quad (2)$$

- Определяне на интервалите на вариране.

$$\lambda_i = \frac{x_{imax} - x_{imin}}{2} \quad (3)$$

- Натуралните стойности на факторите t_i се привеждат (кодират) в относителен вид.

$$x_i = \frac{t_i - x_{i0}}{\lambda_i} \quad (4)$$

- Построяване на разширена матрица с кодирани фактори F с използване на резултатите от (2), (3) и (4).

x_0	x_1	x_2	x_1x_2	x_1^2	x_2^2
1	-1	-1	1	1	1
1	1	0	0	1	0
1	1	0	0	0	0
1	1	0	0	0	0
1	1	0	0	1	0
1	-1	0	0	1	0
1	-1	1	-1	1	1

- Построяване на транспонирана матрица F^T.

1	1	1	1	1	1	1
-1	1	1	1	1	-1	-1
-1	0	0	0	0	0	1
1	0	0	0	0	0	-1
1	1	0	0	1	1	1
1	0	0	0	0	0	1

- Построяване на информационна матрица $F_T F$.

7	0	-1	-1	5	2
0	5	-1	0	-1	-2
-1	-1	2	-2	0	0
-1	0	-2	2	-1	0
5	-1	0	-1	4	2
2	-2	0	0	2	2

- Построяване на ковариационна матрица $(F_T F)^{-1}$.

1.90	0.96	4.41	4.44	-1.23	0.33
0.96	2.43	7.22	7.41	0.58	1.11
4.41	7.22	26.18	26.73	1.64	2.05
4.44	7.41	26.73	27.85	1.92	1.99
-1.23	0.58	1.64	1.92	2.78	-0.74
0.33	1.11	2.05	1.99	-0.74	2.02

- Изчисляване на регресионните коефициенти.

Векторът с оценките на регресионните коефициенти се получава чрез матрично решение на система линейни уравнения от вида:

$$b = (F^T F)^{-1} F^T y \quad (5)$$

След решаване на системата (5) се получават следните оценки на коефициентите:
$b_0 = 84.72, b_1 = -14.79, b_2 = -29.34, b_{12} = -132.27, b_{11} = 7.38, b_{22} = 21.18.$

Видът на търсената регресионна зависимост е:

$$y = 84.72 - 14.79x_1 - 29.34x_2 - 132.27x_1x_2 + 7.38x_1^2 + 21.18x_2^2 \quad (6)$$

2. Проверка за адекватност на получения модел [3, 4].

Определят се прогнозите $y^{\sim}$ и техните отклонения ε от регресионния модел:

x_0	x_1	x_2	x_1x_2	x_1^2	x_2^2	y	$y^{\sim}$	ε	ε^2
1	-1	-1	1	1	1	32	32	0	0
1	1	0	0	1	0	118	117	1	0.47
1	1	0	0	0	0	91	91	0	0.05
1	1	0	0	0	0	110	109	1	2.09
1	1	0	0	1	0	113	115	-2	3.51
1	-1	0	0	1	0	110	110	0	0
1	-1	1	-1	1	1	231	231	0	0

Сумата от квадратите на отклоненията е $Q_{ост}$ = 6.12.

Проведени са десет допълнителни измервания за $x_1 = 32$, $x_2 = 20$:

y_{e1}	y_{e2}	y_{e3}	y_{e4}	y_{e5}	y_{e6}	y_{e7}	y_{e8}	y_{e9}	y_{e10}
118	120	119	117	120	116	118	117	115	115

Изчисляват се статистически оценки на допълнителните данни:
средна стойност - 117.5, дисперсия s_{yc}^2 - 3.39.

Определят се степените на свобода на допълнителния експеримент:
$v_c = N_c - 1 = 9$, където $N_c = 10$ - брой допълнителни опити.

Определят се степените на свобода за модела:
$N = 7$ - брой опити, $k = 6$ - брой коефициенти, $v_{mod} = N - k = 1$ - степени на свобода за модела.

Дисперсията за предложения модел е $s_{mod}^2 = \frac{Q_{ост}}{v_{mod}} = 6.12$.

Изпълнява се тестване на модела по критерий на Фишер. При ниво на значимост $\alpha = 0.05$ критичната стойност е $F_{крит}(\alpha, v_{mod}, v_c) = 5.12$.

Числото на Фишер за модела е $F = \frac{s_{mod}^2}{s_{yc}^2} = 1.81$.

Сравнението на числото на Фишер с критичната стойност показва, че $F < F_{крит}$. Следователно предложеният модел (6) е адекватен.

III. Заключение

Представеният подход показва съществуването на статистически обоснована математическа зависимост на времето за оценка на финансов обект и съставящите го инструменти. Следователно е възможно чрез провеждане на неголям брой експерименти да бъдат предсказани времената за оценяване на голям брой обекти, изграждащи изчислителна финансова система. Такъв подход за предсказване на времена би могъл да се приложи за реални финансови обекти, съставени от повече инструменти, върху които се изпълняват различни по сложност симулационни анализи.

Литература:

[1]. Penev I., A. Antonov, Realization of Portfolio Management System in a distributed computing environment, International Conference Computer Science. Sofia, 2009

[2]. Penev. I., A strategy for management of parallel jobs with concurrent access to a common data source in a distributed computing environment, Informatics in Scientific Knowledge. Varna Free University, 2010

[3]. Майзер Х, Н. Ейджин, Р. Тролл и др. Исследование операций. Методологические основьі и математические методьі, Мир. Москва, 1981

[4]. Калиткин Н.Н., Численньіе методьі, Наука. Москва, 1978

Авторът изказва благодарност към доц. д-р инж. Анатолий Антонов за ценните съвети, получени при подготовката на материала.

За контакти:
ас. Ивайло Пенев
катедра „Компютърни науки и технологии”
Технически Университет - Варна
E-mail: ivailopenev@yahoo.com

Рецензент: доц. д-р инж. В. Николов, ТУ – Варна

СРЕДА ЗА СЪГЛАСУВАНЕ НА МРЕЖА ЗА ОБМЕН С ИЗТОЧНИЦИ НА СВРЪХ ГОЛЕМИ И НЕПРЕСТАННО НАРАСТВАЩИ ОБЕМИ ДАННИ

Ваня С. Топалова, Радослав П. Райчев

Резюме: В работата се представя подход, методика и алгоритъм за проектиране на система за управление и трансфер в мрежова среда на архиви от мултимедийни данни със свръх голям и непрестанно нарастващ обем. Предложеният алгоритъм, внедрен в практиката, намалява риска от повторно заемане на мрежови ресурси и от прекъсване по таймаут.

Method and Algorithm for Equalizing Huge Volume Data Sources with the Network

Vanya S. Topalova, Radoslav P. Raychev

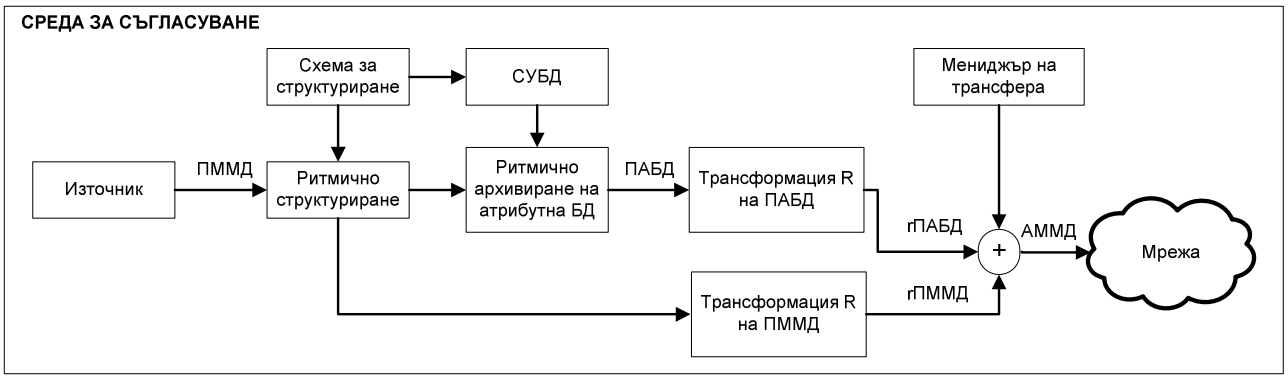
Abstract: This paper describes a design approach, method and algorithm for system, which manages the transfer via network of huge, growing in real-time, multimedia archives. The risks of repetitive reservations of network resources and of session closing on timeout are reduced.

1. Увод

Голям брой потребители генерират потоци от мултимедийни данни (ПММД) с непрестанно нарастващ общ обем и определена ритмичност (месечна, седмична, дневна и т. н.) от системи за видео наблюдение, за контрол на достъпа и за управление и трансфер на архиви в мрежова среда и други. Като правило, ПММД се съпътстват от обслужващи ги атрибутни бази данни (БД). Появата на всеки пореден елемент от ПММД налага предаване на него самия еднократно и – тук е затруднението – ново препредаване на целия, вече увеличил се актуализиран обем на архивите на атрибутната БД.

Проблемът има неоднозначно решение. Стартирането на подобни транзакции в критични часови интервали има негативно влияние и за всички останали мрежови процеси, протичащи паралелно с тях.

Необходимо е да се търсят решения за оптимално използване на пропускателната способност на мрежата чрез прилагане на различни схеми за регулиране на потока от архивите на атрибутната БД (ПАБД) така, че пълният поток АММД = rПММД + rПАБД да бъде получен с минимални загуби от приемника, където r символизира оператор за трансформация със структуриране и компактиране на съответните потоци (фиг.1) [1,2,3,4,5].



Фиг. 1. Среда за съгласуване на архиви от мултимедийни данни

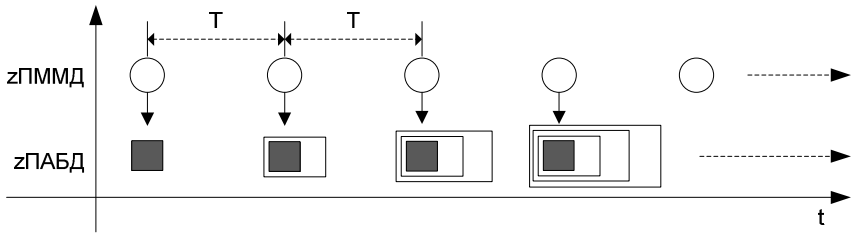
2. Подход за управление и трансфер на архиви от мултимедийни данни

Тук ще бъдат разгледани по-подробно проблемите за съгласуване на потоците на входа на мрежата. На преден план изникват методически проблеми по определяне на ритъма и обемите на въвежданите в мрежата данни при зададени изисквания от страна на крайния потребител.

Потребителските изисквания могат да се определят с поносимия праг на щетите от загуба на **p** записа поради случайно изтриване на данни от атрибутната БД, софтуерен проблем, хардуерен отказ и др. При такава постановка, ритъмът за генериране на данните АММД е функция от прага **p**.

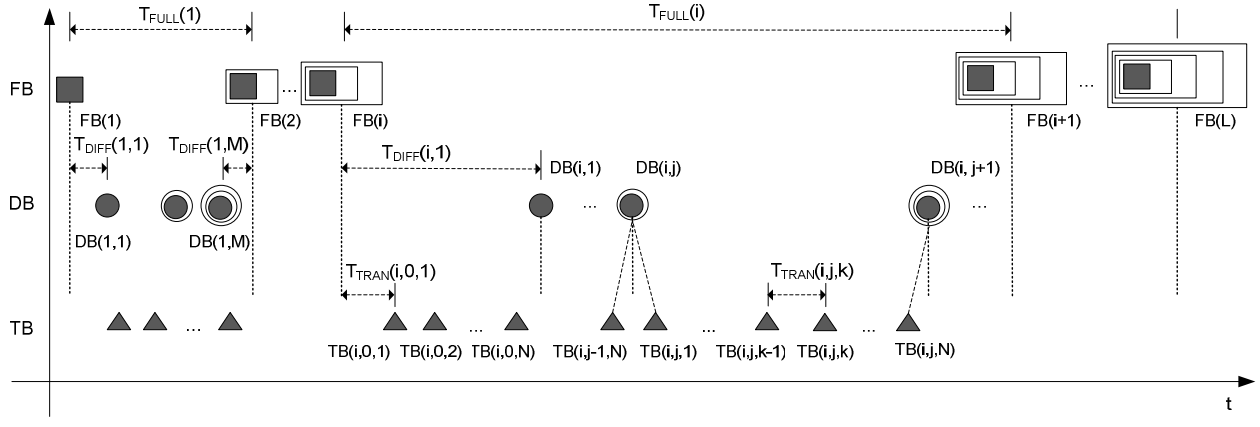
По-често ритъмът за генериране на данни АММД се подчинява на ритъма на потока ПММД, следващ денонощен, седмичен, месечен, … календарен график. Той може да бъде съгласуван с изискванията на конкурентни мрежови процеси по критични часови интервали.

Класическият подход за предаване на rПАБД се състои в изпращане на пълен архив FB (Full Backup) на атрибутната БД в необходимия момент от календарния график (фиг. 2).



Фиг. 2. Класическа схема на график за трансфер на архиви от мултимедийни данни

Предлаганият подход за управление и трансфер на АММД цели оптимизиране на обема на потока rПАБД. Систематичното предаване на пълния архив на атрибутната БД по фиг. 2 се заменя с предаване на три типа архиви по схема, наречена F-D-T по имената на предаваните архиви (фиг. 3): пълни FB (Full Backup), диференциални DB (Differential Backup) и транзакционни TB (Transactional Backup) с периодичности $\mathbf{T_{FULL}}$, $\mathbf{T_{DIFF}}$ и $\mathbf{T_{TRAN}}$ [1,2,3,5].



Фиг. 3. Предлагана схема за трансфер на архиви от мултимедийни данни

Пълният архив на атрибутната база FB е самодостатъчен за възстановяване на данните в приемната дестинация, но е неефикасен за трансфер през мрежата, т. к. обемът му е непрестанно нарастващ (фиг. 2). Смисълът на предложения подход е да намали честотата на

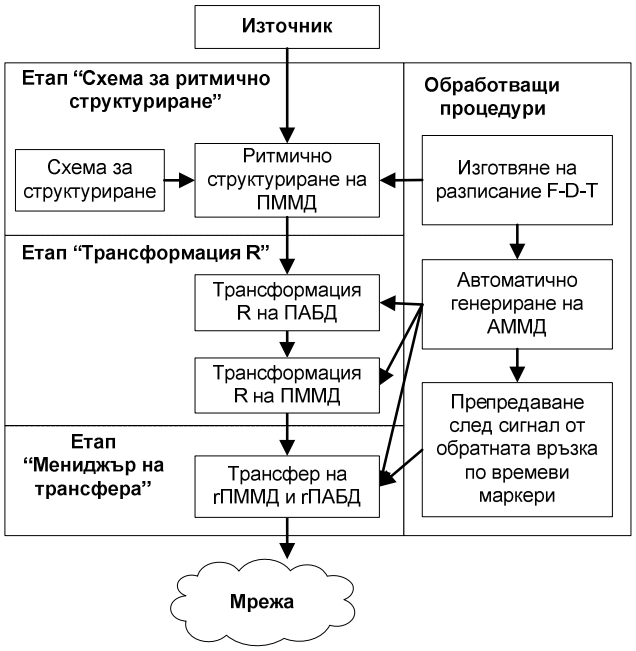
предаване на FB. Всеки интервал ($\mathbf{T_{FULL}}$) съдържа само един архив FB, предаван в първата мрежова сесия от интервала. Архивът **FB(i)** в текущия интервал $\mathbf{T_{FULL}}$ **(i)** (**i** е номера на интервала) обезсмисля всички предходни архиви FB. Актуализацията на **FB(i)** се осигурява до края на $\mathbf{T_{FULL}}$ **(i)** с кумулативни типове архиви DB и TB през следващите мрежови сесии от $\mathbf{T_{FULL}}$ **(i)**.

Диференциалните архиви **DB(i,j)**, **j=[1,M]**, съдържат модификациите, направени в атрибутната БД след **FB(i)**. Всеки следващ диференциален архив **DB(i,j)**, **j=[2,M]**, съдържа в себе си предходния **DB(i,j-1)** и всички TB след него. Когато обемът на **DB** стане съизмерим с обема на пълния архив FB, създаването на DB се замества със следващия пълен **FB(i+1)**.

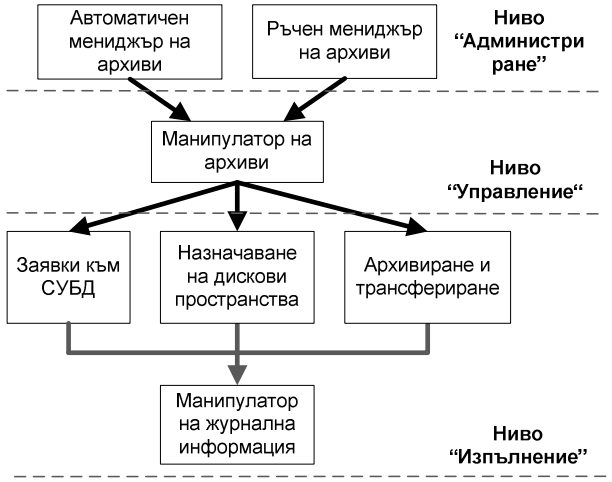
Транзакционните архиви TB също облекчават изискванията по заемане на мрежови ресурси. Всеки транзакционен архив **TB(i,j,k)**, **k=[1,N]**, съдържа потвърдените промени в атрибутната БД в интервала $\mathbf{T_{TRAN}}$ след генериране на предхождащия го архив (независимо от неговия тип – FB, DB, TB).

На пръв поглед изглежда (фиг. 3), че предложеният подход увеличава, а не намалява общия обем на потока rПАБД. Но това не е така, защото периодът $\mathbf{T_{FULL}}$ е много по-голям от аналога си **T** по фиг. 2, поради това, че $\mathbf{T_{TRAN}=T}$. От друга страна, невъзможността да се предаде един архив DB или TB осуетява успешното разархивиране на атрибутната БД до периода $\mathbf{T_{TRAN}(i,j,k)}$ и ограничава опциите за възстановяване на базата или до преходния период $\mathbf{T_{TRAN}(i,j,k-1)}$, или до момента на успешното му предаване.

Основните етапи в подхода са три (фиг. 4).



Фиг. 4. Функционален модел на средата



Фиг. 5. Управление на средата

На етап „Схема за ритмично структуриране" ПММД се обособяват в атомарни единици, като се вземат под внимание както характеристиките на генерираните от източника потоци от данни, така и общите цели и ограничения, поставени пред системата за трансфер. Най-често ПММД се организират в директории по часове, дни, месеци, години и т.н, докато структурата на ПАБД се задава от избраната схема за архивиране на атрибутната БД.

На етап „Трансформация R" потоците ПАБД и ПММД преминават през оператора **r** за трансформация с цел структуриране и компактиране. Компресията на архивите води до заемане на по-тясна честотна лента при трансфер на АММД в мрежата.

На етап „Мениджър на трансфера" се задават конфигурационните параметри за преноса по мрежата и при евентуален неуспех се изпълнява процедура за повторно препредаване на

АММД. Тук може да се варира с максималните размери на атомарната единица за архивите на всеки поток rПАБД и rПММД и на целия архив. Оптимизацията на обема на обединения поток АММД се извършва основно в предходните два етапа.

3. Методика за управление на трансфера на мултимедийни данни със свръх голям обем

Предложената методика за управление на трансфера на ПММД със свръх голям обем е структурирана на три нива (фиг. 5). Ниво „Администриране" инициира обработващите процедури. Моментите на тяхното изпълнение следват графика на автоматичния мениджър на архиви до намесата на ръчния мениджър на архиви. Ниво „Управление" е алгоритмизирано в точка 4. Операторите от ниво „Изпълнение" осъществяват обработки на обслужваните от тях ресурси СУБД, БД, снимков материал, дискови пространства, журнали.

Основните моменти в предложената методика са следните.

1. Анализ на характера на ПММД, на зависимостите между отделните логически фрагменти, на изискванията на мрежовата среда – честотна лента, времеви интервали на мрежово натоварване и на приоритетите на различните типове ММД.

Избор на архивираща процедура за формиране на rПАБД.

2. Избор на стратегия за архивиране на атрибутната БД.
3. Избор на схема за формиране на rПАБД – комбинация от типовете архиви FB, DB и TB и на календарен график за предаване на rПАБД.

Избор на архивираща процедура за формиране на rПММД.

4. Избор на схема за формиране на rПММД в съответствие със структурната им организация (най-често структурирани в часови и календарни директории).

Избор на архивираща процедура за обединения поток АММД.

5. Съгласуване на процедурите за архивиране и трансфер по мрежата в обща архивираща процедура за обединения поток АММД.

Избор на механизъм за препредаване на АММД.

6. Избор на механизъм за препредаване на обединения поток АММД съгласно обратна връзка по времеви маркери за следене на трансфера на АММД.

4. Алгоритъм за управление и трансфер на мултимедийни данни със свръх голям обем

Алгоритъмът за управление и трансфер на ММД със свръх голям обем съдържа няколко ключови процедури: (1) процедура за изготвяне на разписание F-D-T, (2) процедура за автоматично генериране на АММД и (3) процедура за евентуално препредаване след сигнал от обратната връзка по времеви маркери.

Процедура за изготвяне на разписание F-D-T - изготвянето на разписание за формиране на rПАБД с три типа архиви - FB, DB и TB, именувано като схема F-D-T, се осъществява на ниво „Администриране" (фиг. 5). Тази процедура задава сценария за генериране на обединения поток АММД. Избраният сценарий се свежда за изпълнение до операторите от по-ниските слоеве в архитектурния модел на системата.

Procedure **Процедура за изготвяне на разписание F-D-T()**
Задаване на конфигурацията()
if (Не съществува архив FB() ||
Съществува архив FB създаден преди интервал >= T_{FULL}())
return Процедура за автоматично формиране на АММД(FB)
else if ((Не съществува архив DB() &&

Съществува архив FB създаден преди интервал >= T_{DIFF}()) ||
Съществува архив DB създаден преди преди интервал >= T_{DIFF}())
return Процедура за автоматично формиране на АММД(DB)
else if (Не съществува архив ТВ() ||
Съществува архив ТВ създаден преди интервал = T_{TRAN}())
return Процедура за автоматично формиране на АММД(ТВ)
else if (Съществува архив ТВ създаден преди интервал >= T_{TRAN}())
return Процедура за автоматично формиране на АММД(DB)
endif
End

Процедура за автоматично генериране на АММД - основна за ниво „Управление" (фиг. 5). Типът архив, който ще бъде генериран, зависи от предложената и внедрена *Процедура за изготвяне на разписание F-D-T()* и от механизма за следене на времевите маркери на потоците rПММД и rПАБД. Особено важна функционалност за успешното предаване по мрежата е процедурата за повторно предаване съгласно обратната връзка по времеви маркери.

Procedure **Процедура за автоматично формиране на АММД(параметри)**
Извличане на текущи настройки(параметри)
if NOT(Успешно изпълнение) **return** "Край"
endif
if NOT(Следенето на времеви маркери е активно()) **return** "Край"
endif
if (Има нужда от резервно копие на архива()) **return**
Осигуряване на ресурси за съхраняване на резервно копие()
endif
Задаване на конфигурацията за създаване на BACKUP архив()
Създаване на BACKUP архив на атрибутната БД()
if (Успешно е изпълнена процедурата за създаване на BACKUP архив())
Изтриване на старите BACKUP архиви()
else return "Край"
endif
Задаване на конфигурацията за архивиране на BACKUP архива()
Архивиране на BACKUP архива и трансфер на rПАБД(макс. размер на архив, макс. размер на единица от архива)
Процедура за препредаване след сигнал от обратната връзка(rПАБД)
Архивиране на ПММД и трансфер на rПММД(макс. размер на архив, макс. размер на единица от архива)
Процедура за препредаване след сигнал от обратната връзка(rПММД)
Актуализиране на по-старите архиви rПАБД по схема F-D-T
End

Procedure **Процедура за препредаване след сигнал от обратната връзка(параметри)**
Извличане на текущи настройки(параметри)
Препредаване след сигнал от обратната връзка()
if NOT(Успешно изпълнение) **return** "Край"
endif
Актуализиране на времевите маркери()
if (Има нужда от резервно копие() &&
Има достатъчно свободно място на дисковото пространство())
return Създаване на резервно копие()

endif

End

5. Дискусия

Предложеният подход за съгласуване на източници на свръх големи и непрестанно нарастващи обеми данни с мрежата осигурява възможността за възстановяване на атрибутната БД след срив при намален обем на предавания поток rПАБД. Това наблюдение е вярно при реалистичното предположение, че периодите **T** и $\mathbf{T_{TRAN}}$ (от фиг. 2 и фиг. 3 съответно), са равни.

Наистина той позволява атрибутната БД да се възстанови във всеки момент, като се използва малък брой от последните предадени/получени архиви FB, DB и TB. Ако атрибутната БД трябва да се възстанови в момент

$$t_0=\sum_{s=1}^{i} t_{FULL}(s) + \sum_{u=0}^{j} t_{DIFF}(i,u) + \sum_{v=1}^{k} t_{TRAN}(i,j,v),$$

то вместо общият обем TotalVolume(t_0) на архивите rПАБД предаден до момент $\mathbf{t_0}$

$$\sum_{s=1}^{i-1} FB(s) + FB(i) + \sum_{s=1}^{i-1}\sum_{u=0}^{M} DB(s,u)) + \sum_{u=1}^{j-1} DB(i,u) + DB(i,j) + \sum_{s=1}^{i-1}\sum_{u=0}^{M}\sum_{v=1}^{N} TB(s,u,v) + \sum_{u=0}^{j}\sum_{v=1}^{N} TB(i,u,v) + \sum_{v=1}^{k} TB(i,j,v)$$

функцията за разархивиране на атрибутната БД ще използва само обем FDTVolume(t_0)

$$FB(i) + DB(i,j) + \sum_{v=1}^{k} TB(i,j,v),$$

който е по-малък за всяко t_0, защото съдържа само втората, петата и осмата съставляващи от TotalVolume(t_0).

6. Заключение

Предложеният подход за съгласуване на мрежа за обмен с източници на свръх големи и непрестанно нарастващи обеми мултимедийни данни минимизира обема на предаваните мрежови потоци, осигурява интегритет на данните и намалява риска от повторно заемане на мрежови ресурси и от прекъсване по таймаут. Основните му принципи са заложени в реализирана за мултимедийни данни програмна среда, приложима и за други типове структурирани данни, обемите на които, заедно с тези на обслужващите ги атрибутни бази данни, нарастват непрекъснато. Описаните методика и алгоритъм са тествани и внедрени в реално функционираща софтуерна система.

Литература

[1]. Dewson, R. Beginning SQL Server 2005 for Developers. Apress, 2006.

[2]. Mistry, R., C. Amaris, A. Minty. SQL Server 2005 Management and administration, Sams, 2007.

[3]. Rizzo, T., A. Machanic, J. Skinner et al. Pro SQL Server 2005. Apress, 2006.

[4]. Tucker, N. Disaster recovery techniques for SQL Server, GlobalKnowledge, 2009, http://images.globalknowledge.com/wwwimages/whitepaperpdf/WP_Tucker_DisasterRecovery_P.pdf.

[5]. Whalen, E., M. Garcia, B. Patel et al. Microsoft SQL Server 2005 Administrator's Companion, Microsoft Press, 2006.

За контакти:
инж. Ваня Топалова
катедра „Компютърни системи и технологии”
Технически университет – Габрово
E-mail:vtopalova@tugab.bg

доц. Радослав Райчев
катедра „Компютърни системи и технологии”
Технически университет – Габрово
E-mail:raychev@tugab.bg
Рецензент: доц. д-р инж. П. Антонов, ТУ – Варна

Bulgaria
Communications
Chapter

ПОВЫШЕНИЕ ДОСТОВЕРНОСТИ РАБОЧЕГО ДИАГНОСТИРОВАНИЯ В ОБРАБОТКЕ ПРИБЛИЖЕННЫХ ДАННЫХ

Александр В. Дрозд

Резюме: Исследуется достоверность методов рабочего диагностирования вычислительных устройств. Отмечается рост значимости обработки приближенных данных и ее особенности, снижающие достоверность традиционных методов рабочего диагностирования в контроле приближенных результатов. Рассматриваются пути повышения достоверности методов рабочего диагностирования и выделяется подход с использованием сокращенных операций. Исследуются особенности сокращенных операций в обработке мантисс. Предлагается метод контроля по модулю сокращенной операции в матричном умножителе.

Development Stages of On-Line Testing in Computing Devices

Alexander V. Drozd

Abstract: Reliability of the on-line testing methods in computing devices is investigated. Growth of the importance in the approximate data processing and its features reducing reliability of the traditional on-line testing methods in checking the approximated results are noted. The ways to increase reliability of on-line testing is considered and an approach with use of truncated operation is selected. Te features of the truncated operation in mantissas processing is researched. A method of residue checking the truncated operation in iterative array multiplier is offered.

1. Введение

Рабочее диагностирование (РД) поддерживает работу цифровых схем компонентов компьютерных систем, реализуя функции контроля при выполнении основных операций на фактических данных [1].

Изначально РД формировалось, наследуя методы и средства помехоустойчивого кодирования, которые складывались в теории и практике связи для противостояния помехам при передачи данных на длинные расстояния. К таким средствам относятся кодеры и декодеры, обеспечивающие кодирование данных помехоустойчивым кодом при их передаче и декодирование при приеме. В процессе передачи данных кодеры и декодеры считались абсолютно надежными, что также было унаследовано РД и перенесено на его средства – схемы контроля, которые не проверялись на фактических данных [2].

С 1968 г. начинается новый этап в эволюции РД, который знаменуется становлением и развитием теории и практики построения самопроверяемых цифровых схем. В определениях полностью самопроверяемой схемы (защищенной и самотестируемой в заданном классе неисправностей) формулируется основное требование, предъявляемое к методам РД – обнаруживать неисправность цифровой схемы по первой ошибке результата, что для самого РД определяет цель – обнаружение неисправностей, т.е. контроль аппаратуры на ее исправность [3].

За прошедший период времени были разработаны методы построения самопроверяемых схем практически для всех типов цифровых устройств, используя различные подходы и способы кодирования обрабатываемых данных. Уже к концу 90-х годов можно отметить существенное сокращение количества научных статей по РД (on-line testing), что можно было бы объяснить успехами, достигнутыми в этом направлении. Действительно, были получены решения, обеспечивающие обнаружение по первой ошибке всех наиболее вероятных неисправностей

цифровых схем: асинхронных и синхронных управляющих автоматов с памятью и комбинационных схем, запоминающих и вычислительных устройств (ВУ) [4]. Методы РД, обеспечивающие построение самопроверяемых схем стали традиционными.

Следует отметить, что теория построения самопроверяемых схем стала развиваться в условиях, когда еще не оформилось разделение цифровых устройств на управляющие и ВУ. В основе построения их схем лежали единые методы формального синтеза функций, что находило отражение и в единстве подходов к РД. Поэтому определения самопроверяемых схем также не разделяют цифровые схемы на управляющие и вычислительные, принимая за основу управляющие схемы, особенностью которых является обработка точных, т.е. целых по своей природе нумерованных данных – управляющих слов. Основываясь на таком фундаменте, теория и практика построения самопроверяемых схем оказалась в плену модели точных данных, когда все данные априори рассматриваются как точные, игнорируя их настоящую природу.

Цифровые схемы ВУ обрабатывают числовые данные, которые, как правило, являются результатами измерений или их обработки, т.е. приближенными данными.

Высвобождаясь от плена модели точных данных, следует отметить, что настоящая цель РД состоит в контроле достоверности результатов, вычисляемых на цифровых схемах в процессе обработки фактических данных. Цель РД, следующая из теории построения самопроверяемых схем, в общем случае не выдерживает критики [5], однако не противоречит настоящей цели для точных данных. Действительно, ошибка, обнаруженная средствами РД одновременно укажет и на наличие неисправности в цифровой схеме и на недостоверный результат. Однако ошибочный результат тождественен недостоверному только в случае точных данных.

Таким образом, теория и практика построения самопроверяемых схем, сыгравшая в становлении РД безусловно важную положительную роль, ограничивает развитие РД ВУ рамками обработки точных данных. В условиях постоянного роста значимости компьютерной обработки приближенных данных сложившиеся рамки приводят к недопустимому отставанию РД от требований, предъявляемых развитием объектов РД.

В следующих разделах статьи раскрываются особенности обработки приближенных данных, влияющие на достоверность методов РД, и один из путей повышения достоверности на примере контроля по модулю сокращенного умножения в однотактных ВУ.

2. Особенности обработки приближенных данных

Обработка приближенных данных определяет результат, состоящим из старших верных и младших неверных разрядов [6]. В них неисправности цифровой схемы вызывают ошибки, являющиеся соответственно существенными и несущественными для достоверности результата.

Объект РД характеризуется вероятностью P_C существенной ошибки, под которой далее будем понимать вероятность того, что возникшая ошибка является существенной.

Метод РД характеризуется вероятностью P_O обнаружения ошибки.

Достоверность методов РД, согласно настоящей цели, состоит в контроле достоверности результатов вычислений, что с участием вероятностей P_C и P_O может быть оценено по следующей формуле [7]:

$$D = P_O P_C + (1 - P_O)(1 - P_C). \quad (1)$$

Для точных данных любая ошибка является существенной, т.е. $P_C = 1$, что определяет достоверность контроля результатов $D = P_O$. В полностью самопроверяемых схемах ошибки, вызываемые неисправностями из заданного класса, обнаруживаются с вероятностью $P_O = 1$, из чего следует известная достоверность $D = 1$ традиционных методов РД.

Для приближенных данных достоверность традиционных методов РД с учетом $P_O = 1$ составляет $D = P_C$., что требует оценки вероятности существенной ошибки, которая определяется особенностями приближенных вычислений.

Первой особенностью является использование операции умножения в самой записи приближенного данного. В форматах с плавающей точкой число записывается в виде произведения $m \cdot q^p$, где m – мантисса, q – основание системы счисления, p – порядок [8]. Полное произведение двух операндов удваивает длину результата двуместной операции. Согласно теории ошибок, количество верных разрядов результата не превышает количество верных разрядов операнда, что определяет $P_C \le 0{,}5$. Младшие неверные разряды результата, как правило, отбрасываются, и результаты наследуют разрядность операндов.

Та же одинарная точность результатов может быть достигнута при выполнении сокращенных арифметических операций, где вычисляется усеченный результат, почти вдвое более короткий, чем полное произведение, что допускает $P_C < 1$.

Для согласования действий над мантиссами и порядками в обработке чисел с плавающей точкой выполняются операции нормализации и денормализации мантисс. Денормализация мантиссы выполняется ее арифметическим сдвигом вправо с потерей младших разрядов, выдвигаемых за пределы разрядной сетки. Соответственно теряются существенные ошибки, которые накапливались в результатах всех предыдущих операций, что снижает вероятность P_C. Нормализация мантиссы при вычислении в ней незначащих цифр выполняется циклическим сдвигом влево с заполнением младших позиций неверными разрядами. Это ведет к уменьшению количества верных разрядов и существенных ошибок в них для результатов всех последующих операций, также снижая вероятность P_C [9].

Особенности приближенных вычислений многократно снижают вероятность существенной ошибки P_C и достоверность традиционных методов РД, контролирующих результаты выполнения полных операций.

Таким образом, возможность достигать высокую достоверность $D > 0{,}5$ при высокой вероятности P_C обнаружения ошибки сохраняется лишь в случае выполнения сокращенных арифметических операций, где достигается вероятность $P_C > 0{,}5$.

3. Метод сокращенного умножения

Метод сокращенного умножения основывается на анализе матрицы конъюнкции произведения (МКП) двух нормализованных мантисс $A = A\{1 \div n\}\, 2^{-n}$ и $B = B\{1 \div n\}\, 2^{-n}$ [10].

Матрица разбивается на две части: младшую и старшую. Младшая часть состоит из k младших столбцов, исключаемых из вычислений. По старшей части вычисляется усеченное произведение $V_T = V\{1 \div 2n - k\}\, 2^{-(2n-k)}$, n старших разрядов которого составляют округленный результат $V_O = V\{1 \div n\}\, 2^{-n}$, а младшие $n - k$ разрядов $V_M = V\{n + 1 \div 2n - k\}\, 2^{-(2n-k)}$ отбрасываются. Величина $k = n - log_2 n$ определяется из условия сохранения одинарной точности.

В однотактных ВУ сокращенное умножение почти в два раза упрощает устройство и уменьшает время выполнения операции [11].

В основу метода контроля по модулю сокращенного умножения положено разбиение старшей части МКП на фрагменты [12]

$$V_i = sign\, V_i\, A_i\, B_i, \qquad (1)$$

где $sign\, Vi = +1$, если большая часть конъюнкций расположена в старшей части МКП, и $sign\, Vi = -1$ в противном случае;

$A_i = A\{a_{i1} \div a_{i2}\}\, 2^{-ai2}$ и $B_i = B\{b_{i1} \div b_{i2}\}\, 2^{-bi2}$ – мантиссы множимого A и множителя B (далее – операнды) или их части, содержащие биты от a_{i1} до a_{i2} и от b_{i1} до b_{i2}, соответственно; i — порядковый номер фрагмента Vi.

Минимальное количество фрагментов в разбиении равно $k + 1$ [13]. Это определяет усеченное произведение, как $V_T = \sum_{i=1}^{k+1} V_i$ и его контрольный код по следующей формуле:

$$KV_T = \sum_{i=1}^{k+1} KV_i \,. \qquad (2)$$

Контрольный код фрагмента V_i определяется из (1) как

$$KV_i = Ksign\, V_i\, KA_i\, KB_i, \qquad (3)$$

где $Ksign V_i = sign V_i \bmod m$; $KA_i = A_i \bmod m$; $KB_i = B_i \bmod m$.

Схема контроля однотактного умножителя мантисс с сокращением вычислений показана на рис. 1.

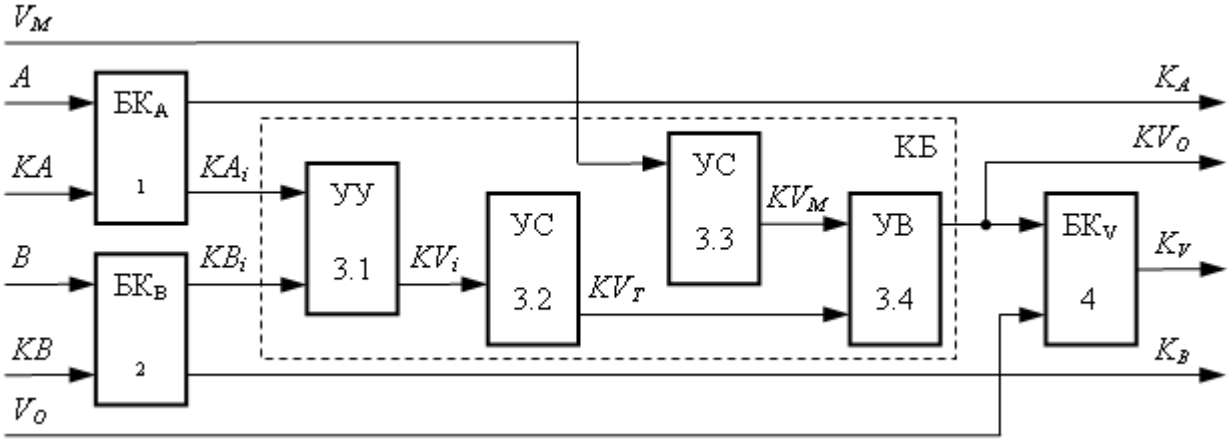


Рис. 1. Схема контроля однотактного умножителя мантисс

Схема контроля содержит блоки контроля операндов $БК_A$ 1 и $БК_B$ 2, а также контрольный блок КБ 3. Блок контроля результата $БК_V$ 4 показан в составе схемы контроля, поскольку формирует код контроля результата. Однако этот блок является одним из блоков контроля $БК_A$ или $БК_B$ схемы контроля для следующего ВУ, где полученный результат является операндом.

Блоки контроля операндов $БК_A$ 1 и $БК_B$ 2 принимают коды сомножителей A и B со входов схемы контроля и сравнивают их по модулю с входными контрольными кодами KA и KB. Результаты сравнения являются кодами контроля операндов K_A, K_B. В процессе проверки также формируются контрольные коды KA_i, KB_i.

Контрольный блок КБ 3 состоит из узла умножения УУ 3.1, узлов сложения УС 3.2 и УС 3.3, а также узла вычитания УВ 3.4. Узел УУ 3.1 по контрольным кодам KA_i и KB_i.вычисляет контрольные коды фрагментов KV_i, согласно формуле (3). Узел УС 3.2 складывает их по формуле (2), определяя контрольный код KV_T усеченного произведения V_T. Узел УС 3.3 формирует контрольный код $KV_M = V_M \bmod m$ отбрасываемых разрядов V_M. Узел УС 3.43 вычисляет контрольный код $KV_O = KV_T - KV_M$ округленного результата V_O.

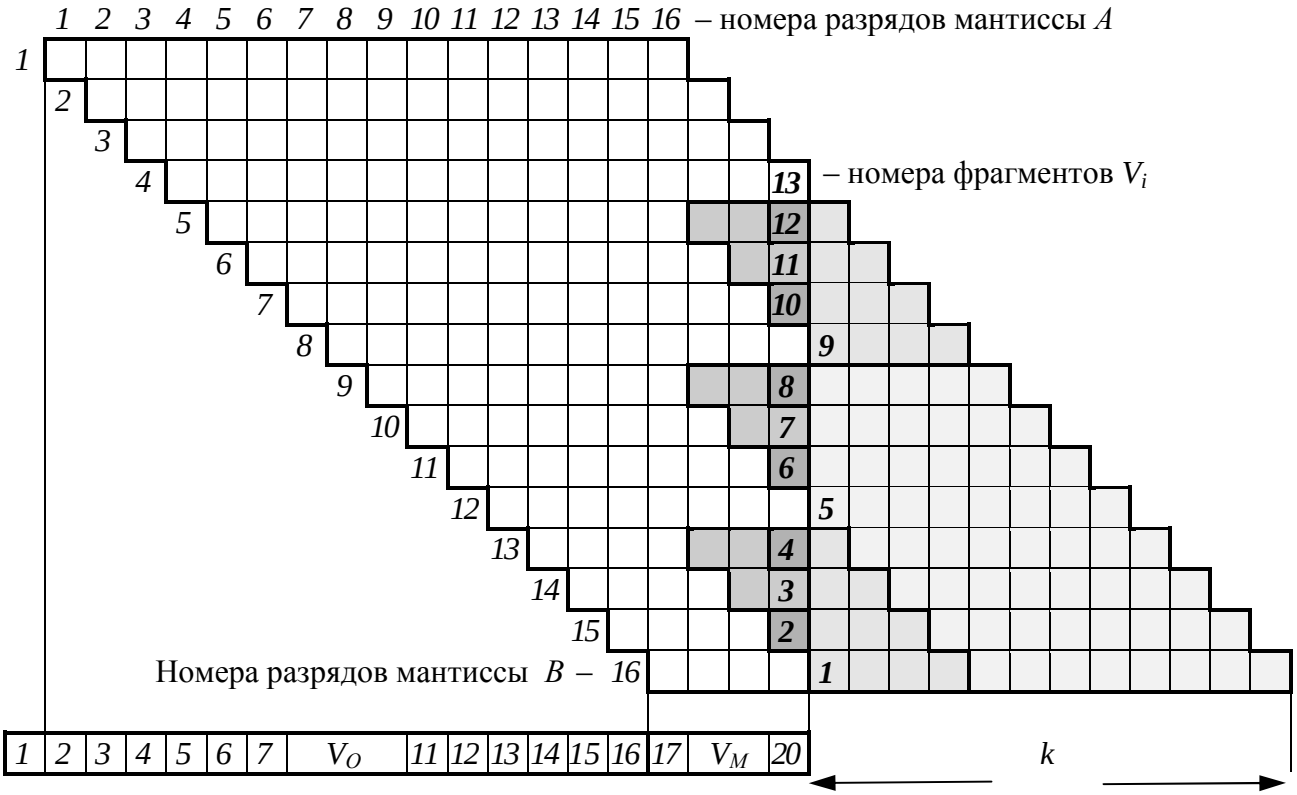
Контрольный код KV_O поступает на выход схемы контроля для сопровождения результата V_O, а также на вход блока контроля результата $БК_V$ 4, который сравнивает с ним по модулю округленный результат V_O и формирует код K_V контроля результата.

На рис. 2 показано разбиение старшей части МКД на фрагменты для $n = 16$ и $k = 12$.

Фрагменты для данного разбиения описываются следующим образом:

$V_1 = -A\{5 \div 8\}\, B\{13 \div 16\}\, 2^{-24}$; $V_2 = A\{5\}\, B\{15\}\, 2^{-20}$;
$V_3 = A\{5, 6\}\, B\{13, 14\}\, 2^{-20}$; $V_4 = A\{7\}\, B\{13\}\, 2^{-20}$;
$V_5 = -A\{9 \div 16\}\, B\{9 \div 16\}\, 2^{-32}$; $V_6 = A\{9\}\, B\{11\}\, 2^{-20}$;
$V_7 = A\{9, 10\}\, B\{9, 10\}\, 2^{-20}$; $V_8 = A\{11\}\, B\{9\}\, 2^{-20}$;
$V_9 = -A\{13 \div 16\}\, B\{5 \div 8\}\, 2^{-24}$; $V_{10} = A\{13\}\, B\{7\}\, 2^{-20}$;
$V_{11} = A\{13, 14\}\, B\{5, 6\}\, 2^{-20}$; $V_{12} = A\{15\}\, B\{5\}\, 2^{-20}$;
$V_{13} = A\{1 \div 16\}\, B\{1 \div 16\}\, 2^{-32}$.

Рис. 2. Разбиение старшей части МКД на фрагменты



Контрольные коды $КА_i$ и $КВ_i$ частей A_i и B_i, длина которых не превышает длины контрольного кода по модулю, непосредственно составляются из разрядов операндов. Для модуля $m = 3$ такими составляемыми являются все контрольные коды $КА_i$ и $КВ_i$ для четных значений i, а также $КА_3$, $КА_7$, $КА_{11}$ и $КВ_3$, $КВ_7$, $КВ_{11}$.

Например, $КА_2 = (A\{5\}\, 2^{-5}) \bmod 3 = -A\{5\}$; $КА_3 = (A\{5, 6\}\, 2^{-6}) \bmod 3 = A\{5, 6\}$.

Остальные контрольные коды $КА_i$ и $КВ_i$ вычисляются в блоках $БК_A$ 1 и $БК_B$ 2 в процессе свертки операндов по следующим последовательностям формул:

$КА_1 = A\{5 \div 8\} \bmod 3$; $КВ_1 = B\{13 \div 16\} \bmod 3$;

$КА_9 = A\{13 \div 16\} \bmod 3$; $КВ_9 = B\{5 \div 8\} \bmod 3$;

$КА_5 = (КА_9 + A\{9 \div 12\}) \bmod 3$; $КВ_5 = (КВ_1 + B\{9 \div 12\}) \bmod 3$;

$КА_{11} = (КА_5 + КА_1 + A\{1 \div 4\}) \bmod 3$. $КВ_{11} = (КВ_5 + КВ_9 + B\{1 \div 4\}) \bmod 3$.

Вероятность существенной ошибки может быть оценена отношением длин округленного результата и усеченного произведения: $P_C = n / (2n - k)$, что с увеличением n стремится к единице, соответственно повышая достоверность контроля приближенных результатов

4. Заключение

Постоянный рост объемов обработки приближенных данных обращает внимание на отставание методов и средств РД от требований, предъявляемых к ним в современных компьютерных системах и компонентах. Традиционные методы РД, сложившиеся в рамках теории и практики построения самопроверяемых цифровых схем, выполняют контроль полных арифметических операций, демонстрируя высокую вероятность обнаружения ошибки. В случае обработки точных данных это обеспечивает соответствующую высокую достоверность контроля результатов. Выполнение полных арифметических операций в приближенных вычислениях определяет низкую вероятность существенной ошибки и такую же низкую достоверность традиционных методов РД, растрачивая высокую вероятность

обнаружения ошибки на выявление наиболее частых несущественных ошибок. Только сокращенные арифметические операции, выполняемые над мантиссами, позволяют достигать высокую вероятность существенной ошибки и использовать методы РД с высокой вероятностью обнаружения ошибки, такие как контроль по модулю, для повышения достоверности контроля приближенных результатов.

Контроль по модулю сокращенного умножения мантисс выполняется путем разбиения обрабатываемой части МКП на фрагменты. Контрольные коды фрагментов являются произведением контрольных кодов частей операндов, составленных из разрядов этих операндов или вычисленных при их проверке. Сумма контрольных кодов фрагментов определяет контрольный код усеченного произведения, а за вычетом отбрасываемых разрядов – контрольный код результата.

Сокращенные операции позволяют не только почти в два раза упрощать однотактные ВУ и снижать время вычислений, но также повышать вероятность существенной ошибки и достоверность контроля приближенных результатов.

Литература

[1]. Пархоменко П. П., Е. С. Согомонян и др., Основы технической диагностики, М. Энергия 1981.

[2]. Кузьмин И.В., В.А Кедрус, Основы теории информации и кодирования. 2-е изд., К. Вища школа 1986.

[3]. Anderson D. A., G. Metze, Design of totally self-checking check circuits for m-out-of-n codes, IEEE Trans. Comput., V. C – 22, N 3, 1977.

[4]. Согомонян Е. С., Е. В. Слабаков, Самопроверяемые вычислительные устройства и системы (обзор), Автоматика и телемеханика. № 11, 1981.

[5]. Drozd A., M. Lobachev, J. Drozd, The problem of on-line testing methods in approximate data processing, Proc. 12th IEEE International On-Line Testing Symposium, Como (Italy), 2006.

[6]. Демидович Б.П., Марон И.А. Основы вычислительной математики, М. Физматгиз 1966.

[7]. Дрозд А. В., Нетрадиционный взгляд на рабочее диагностирование вычислительных устройств, Проблемы управления. № 2, 2008.

[8]. ANSI/IEEE Std 754-1985. IEEE Standard for Binary Floating-Point Arithmetic. IEEE, New York, USA, 1985.

[9]. Мілейко І. Г., М. Б. Копитчук, О. В. Дрозд, Зниження вимог до інформаційної надійності у інформаційно-вимірювальних системах, Електромашинобудування та електрообладнання. № 72, 2009.

[10]. Савельев А. Я., Прикладная теория цифровых автоматов, М. Высшая школа 1987.

[11]. Рабинович З. Л., Раманаускас В. А. Типовые операции в вычислительных машинах, К. Техника 1980.

[12]. Дрозд О. В., Контроль за модулем обчислювальних пристроїв, Одеса АО Бахва 2002.

[13]. Дрозд А. В., Контроль по модулю однотактного умножителя с сокращенным выполнением операции, Электронное моделирование. Том 20. № 3, 1998.

Для контактов:
проф., д-р техн. наук Александр Дрозд
кафедра компьютерных интеллектуальных систем и сетей
Одесский национальный политехнический университет – Одесса
E-mail:drozd@ukr.net

Рецензент: доц. д-р инж. О. Железов, ТУ – Варна

Importance Analysis for Predicting Data Access Behaviour in Object-Oriented Applications

Stoyan Y. Garbatov, João P. Cachopo

Abstract: This work presents an innovative system for analyzing and predicting the behaviour of object-oriented applications, with regards to the data they manipulate. The system is validated by the execution of the oo7 benchmark, which has been modelled as a stochastic process through Monte Carlo simulations. The overheads introduced are in the order of 4%-5%, with regards to the non-instrumented application. The system is sufficiently flexible to be applied to any object-oriented application.

Анализ на значимостта за предвиждане на поведението спрямо достъпа до данни в обектно-ориентирани приложения

Стоян Й. Гърбатов, Жоао П. Кашопо

Резюме: Тази работа представя една иновативна система за анализиране и предсказване на поведението на обектно-ориентирани приложения, по отношение на данните, които те манипулират. Системата е тествана чрез бенчмарка oo7, който е моделиран като стохастичен процес, използвайки Монте Карло симулации. Въведените забавяния, при изпълнението на приложението, са от порядъка на 4% -5%, спрямо не-инструментализираната версия. Системата е достатъчно гъвкава, за да бъде приложена към всяко обектно-ориентирано приложение.

1. Introduction

Most applications need to fetch data from external resources to perform their functions. This is a time consuming operation. Because of this, applications that access large volumes of data must be carefully engineered to have acceptable performance.

The cost of fetching data from an external resource depends on a series of factors. A good design rule is to minimize the number of round-trips made by the application to load the data. On the other hand, it should be also sought out to minimize the amount of data fetched. Consequently, performing a single round-trip that fetches exactly the data that is needed for a given operation would be the optimal solution. A common example of such data-fetching approach is the use of complex SQL queries to retrieve data from multiple tables at once from a relational database. Unfortunately, this idealized goal is seldom possible to achieve and the reasons are manifold.

The current state-of-the-practice is to leave to the programmers the burden of knowing what to fetch and when to do it. However, as applications become more complex and are developed with higher-level languages, knowing beforehand which data is needed for a particular computation, becomes humanly unfeasible.

Moreover, even if the data-fetching patterns for an application are fine-tuned, these adjustments may become useless or even counterproductive when either the requirements for the application or the data structure change, even if slightly.

Additionally, many applications use caches to reduce the need to fetch data from external resources, and, thereby, accelerate the application. Obviously, the use of these caches and their dynamic contents influence the optimal fetching strategy.

The data-fetching strategy for an application should not be a responsibility of the programmers. It should be determined automatically, based on the data-access behaviour observed.

Several related works have been published. Knafla [1] presents a methodology where the main goal is to facilitate the prefetching of persistent objects through the prediction of data access.

The relationships between the objects are modelled by a discrete-time Markov Chain. The work offers an interesting view regarding the use of stochastic models for predicting the behaviour of applications. However, only the actually existing relationships between domain objects are taken under consideration with the model. It may be more interesting to consider the effectively observed data accesses, instead of only the static domain object hierarchy.

Bernstein et al. [2] present a technique for data prefetching. The main concept behind it is to associate a context to every object at the time it is loaded. This subsequently allows using the context of the object and the concrete portion of an object's state that is loaded to interpolate about the probability of the application needing to manipulate the same portion of state of other objects sharing that context. That enables the loading of data that would be needed by the application in a pre-emptive fashion (prefetching it), effectively reducing the time lost while waiting for the data to be loaded on demand.

The work presented here is a continuation of the one developed in [3], where the analysis is based on the Bayesian inference model, and is also a part of [4] and [5].

This paper deals with the problem of determining what the behaviour of an application is, with regards to the data accesses that it performs. It describes the design and implementation of a system that predicts the data-access patterns for a Java application. It captures all of the data accesses made by the application during its execution, and uses Importance Analysis model over the collected data to predict future data accesses.

The system was designed to integrate seamlessly with the development of a Java application, and requires minimal effort from the programmer. Additionally, the system is efficient both in the amount of memory that it requires and in the performance overheads that it introduces.

2. System Description

The system is composed of three modules: a code injection module, a data acquisition module, and a data analysis module. The code injection module is responsible for injecting code into the target applications. This code is necessary for the invocation of functionality present in the other two system modules and its injection is performed in a completely automatic fashion by the system to avoid the need for the application programmers themselves to perform any modifications whatsoever to their applications. The second module deals with collecting data about the actual behaviour of the target application, with regards to the data accesses that are performed. Finally, the data analysis module is responsible for applying the Importance Analysis model over the acquired data and, subsequently, for generating the prediction about the most likely behaviour to be observed by the target application, in terms of the data it will access.

2.1. Code Injection

The data-access patterns of an application describe which types of domain objects and which of their fields are accessed during its execution. Consequently, to capture these patterns, it is necessary to identify all the points where data is accessed within the application and instrument them. This allows the subsequent recording of those accesses during the execution of the application.

The system performs this instrumentation at compile-time by means of bytecode rewriting. Consequently, the system performs all the necessary modifications to the target application in an automatic way, without the intervention of the programmer. The modifications are achieved through two instrumentation phases, both of which make use of the Javassist library to inject the code.

The first instrumentation phase injects the code necessary for the identification of the contexts within which all data accesses take place. The context is a key concept in the system. It defines the scope within which all data manipulations take place. For the work presented here, the

context corresponds to the method in which the data accesses take place, but it can be defined in a different way, if the situation so demands (for instance, the context can correspond to the execution of a given service or even a sequence of method invocations that precede the current point of execution).

To identify the contexts at runtime, the first instrumentation phase modifies each method of the application. It injects code that updates the context information associated with the method, upon entering it, and code that cleans the same context information when returning from it.

The second instrumentation phase replaces every field access that exists within the application with the invocation of a previously injected static method. There is a distinct method for each of the fields for every class of the application. These methods determine the surrounding context in which the field is accessed and update the associated statistical information. This is done according to the type of access performed: a distinct method is invoked whether the field is read or written.

Once these two phases are complete, the application can be put into operation, and it will automatically collect the statistics about each data access, without any further modifications.

An important aspect of the solution presented here is the data structure used to store the statistical data. During the instrumentation phases, the system performs an analysis of the structure of the target application via reflection. As a result of this analysis, it generates a set of *PClass* instances to model the structure of each of the target application classes. Each of these *PClass* instances contains a set of *PField* instances, which are used to represent each of the fields that exist in the application class. The information kept in a *PField* covers not only the name and type of a field, but also the number of times that it has been accessed in a context. The subsequent probabilistic analysis uses this information to make its calculations.

The relationship between *PClass* instances and an application class is many-to-one. This can be explained with the fact that the *PClass* instances store information regarding the way that application classes are accessed in the contexts during execution. Consequently, if the same class is used in different contexts, distinct *PClass* instances will keep the information about how that class was used in each.

Taking this into account, it is possible to estimate the upper bounds on the memory requirements for maintaining these structures. The memory will be proportional to the number of classes, times their average number of fields, times the average number of distinct contexts where each class is accessed during the execution. More importantly, note that the memory requirements will not grow continuously, independently of the duration of the execution of the application. In general, this consumption of memory is negligible when compared to the memory needed by the application.

Moreover, given that the probabilistic analysis iterates through the *PField* and *PClass* instances, the time spent in the probabilistic analysis will also take time that is proportional to the previously stated bound.

2.2. Model Implementation

The way how the Importance Analysis model operates is presented here. The data acquisition process, for the model, is decomposed into several steps. While the application is executing, at each field access, the surrounding context is determined. The context has a set of *PClass* instances, which contain the data about what types of accesses have been seen. Once the surrounding context is available, the system performs a lookup over the collection of *PClasses*, resulting in the *PClass* whose field is currently being accessed. After this has been done, the data of how many times the field has been accessed is updated, and the application may proceed with its normal operation.

Based on the Risk Priority Numbers (RPN) technique [6-8] the Importance Analysis model uses three sets of data as input, namely:

• The local probability of a field to be accessed in a given context;
• The global probability of a field being accessed in the application;
• The impact factor, which corresponds to expert judgment of the application developer, indicates how much a given field, is deemed important for the operation of the application.

$$RPN = (L)(G)(I) \quad (1)$$

These three factors are employed as may be seen in Eqn (1), where L is the local access probability, G is the global access probability and I is the impact factor. Once the resulting RPN is calculated, all application fields may be classified and subsequently ordered according to their RPN values. These values reflect the importance of the data, with regards to the effectively observed application behaviour. Based on them, it is possible to make decisions about what are deemed to be the most likely data types to be accessed in a given moment of the application execution and, subsequently, perform optimization techniques that may lead to improving the overall performance. Examples of such techniques are prefetching, and guided caching policies.

3. Results and Evaluation of the System

The oo7 benchmark has been employed for evaluating the system presented here. This benchmark, firstly presented by Carey et al. [9], is often used to evaluate the performance of object-oriented persistence mechanisms. It strives to present a broad set of operations, allowing for the building of a comprehensive performance profile of the system under evaluation.

The benchmark has been designed to boast properties common to different CAD/CAM/CASE applications, although in its details it does not model any specific application. The evaluation is performed through the execution of a series of traversals, updates, and query operations over the underlying object model. The performance metric used is the time that these operations take to execute.

It should be noted that even though there are some random elements present in the functionality performed by the benchmark operations, these are deterministic in their behaviour. In other words, the complete sets of actions which are executed by the oo7 benchmark are predictable. This property makes it somehow inappropriate for evaluating the precision of a system that seeks to predict the behaviour of its target applications.

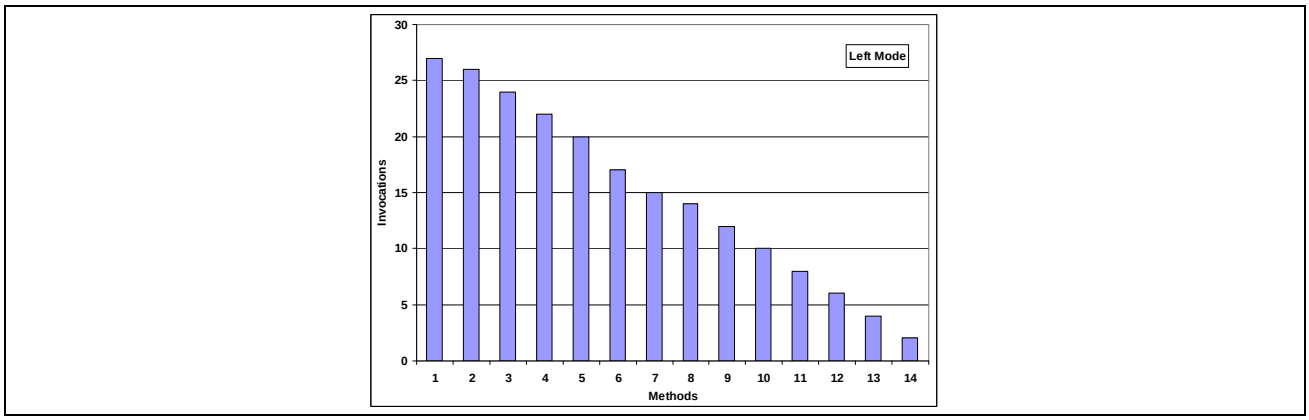


Figure 1 - Method invocations, left mode

With this in consideration, we employed Monte Carlo simulations [10] to make the oo7 benchmark behave like a stochastic process. A stochastic process is one whose behaviour is non-deterministic in that a system's subsequent state is determined both by the process's predictable actions and by a random element. This was done by modelling the number of invocations of the main methods that compose the benchmark. A triangular distribution is used to perform the modelling. This concrete type of distribution is chosen because it is the one most commonly

employed when dealing with a system about which there is little or none information about how it behaves. Three distinct triangular distributions are generated to model the benchmark invocations, namely with left (see Figure 1), middle and right mode location.

The results generated by use of the Importance Analysis model can be observed in Figure 2.

The *z*-axis of the histogram corresponds to the number of fields whose RPN values equal the number obtained by multiplying their associated x and y values. The *x*-axis indicates the local probability access class to which a given field belongs, while the *y*-axis is relative to the global probability access class of the fields.

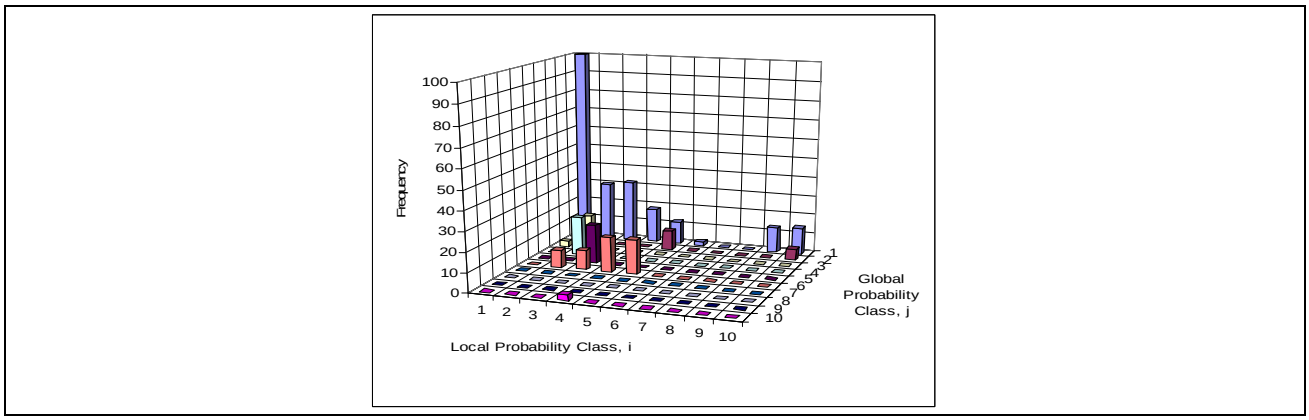


Figure 2 – Importance analysis, left mode

A remark to be made is that most of the application fields belong to the lowest probability classes, while, at the same time, very few of them belong to the higher probability classes. This trend is rather beneficial inasmuch as a smaller data set of likely to be needed data is easier to manage when trying to perform optimizations.

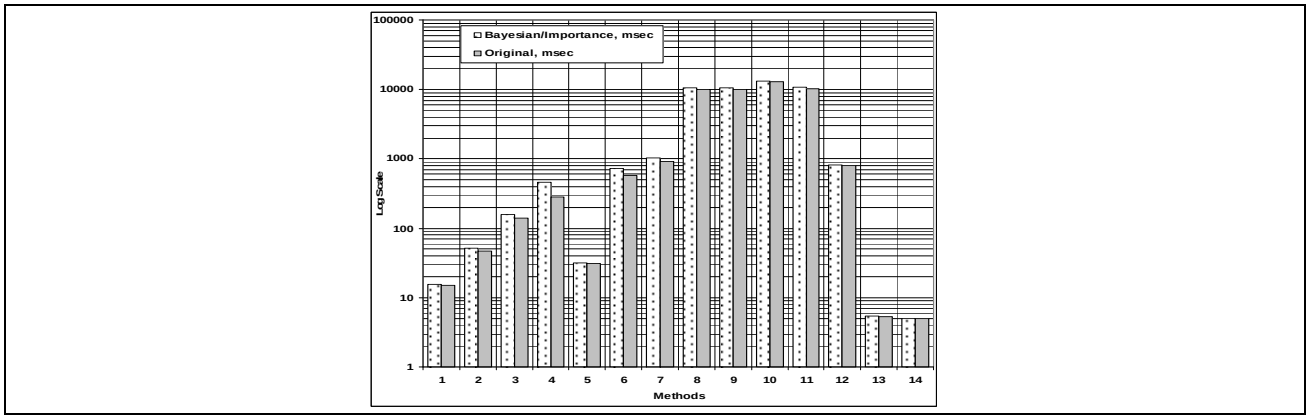


Figure 3 – Method execution times, msec

There is a significant amount of code injected in the application to collect all the information needed to build the models. The need to execute this code causes overheads and penalizes the performance of the application, in comparison with its non-instrumented version. Consequently, it is important to measure those overheads to determine whether they are acceptable in the context of the normal operation of the application. Another aspect to be taken under consideration is the memory space needed for the storage of the statistical data gathered up to a given point in time.

The execution times of the main methods of the OO7 benchmark are summarized in Figure 3 and Figure 4.

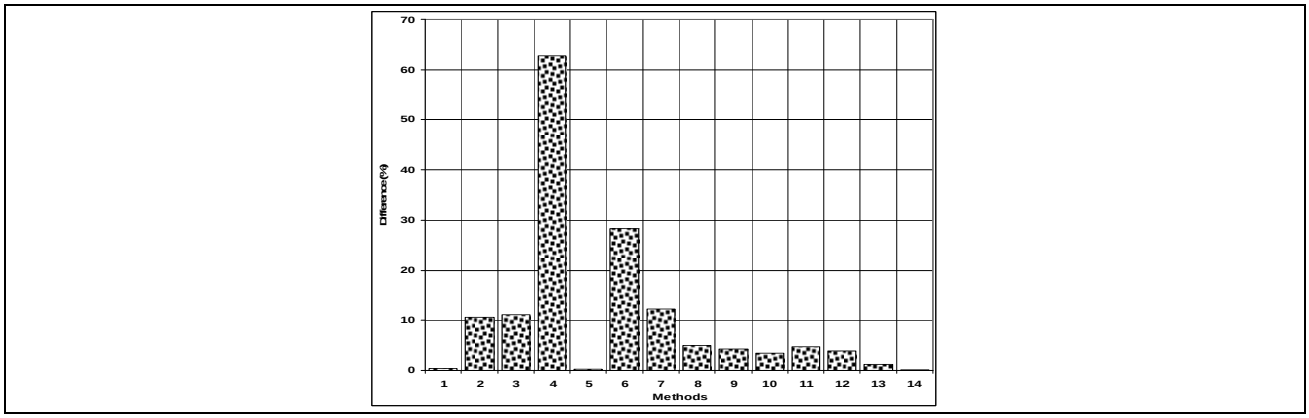


Figure 4 – Difference in execution times, %

In the *x*-axis of the histogram present in Figure 3, we have each of the 14 main methods that define the benchmark. For each of these methods there are two bars: the right one indicates the execution time (in milliseconds) of the method in the original benchmark and the left one is the execution time (in milliseconds) of the instrumented version. The histogram in Figure 4 shows the difference between the bars in percentage. The weighted average, $\overline{overhead}$, is as:

$$\overline{overhead} = \sum_{i=1}^{n} overhead_{method,i} \cdot t_{method,i} \Big/ \sum_{i=1}^{n} t_{method,i} \quad (2)$$

where n is total number of methods, $overhead_{method,i}$ is the overhead, in percentage, associated with method with index i and $t_{method,i}$ is the execution time of method indexed by i. The weighted average of the performance overhead equals 5.15%. As a result of that, the instrumented version is, on average, about 5% slower, in its execution, when compared with the original one. It is deemed that this performance penalty is acceptable.

With regards to the additional memory requirements, due to the storage of the statistical data of observed access patterns, it is not possible to verify any "observable" difference in the memory needed by the original benchmark and the one employing the system developed here. A concrete example would be to say that the statistical data gathered from several executions of the benchmark did not exceed several hundred kilobytes, when saved on the hard disk, while the benchmark process would require a couple of hundred megabytes of memory, when executing. This allows concluding that the memory requirements for the storage of the statistical data necessary for the generation of predictions by the system are negligible.

4. Conclusion

This paper presented a new system for analyzing the behaviour of target applications with regards to the data accesses they perform. The data accesses are analyzed and predicted using advanced probabilistic techniques employing Importance Analysis. Several scenarios were generated and executed, and the subsequent results were analyzed. The system developed for data access behaviour analysis was applied over the oo7 benchmark randomized through the use of Monte Carlo simulation.

The overheads introduced by the system were shown to be in the order of 5%, when comparing the instrumented application with the original one.

The system developed here is flexible enough to be used as a basis for a set of optimization techniques. Special benefits may be obtained when the data being analyzed is persistent. This would allow for the application of optimizations regarding the way that the information is loaded, stored, or cached.

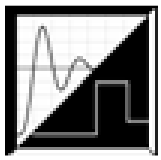
5. Acknowledgements

This work has been performed in the scope of the Pastramy project (PTDC/EIA/72405/ 2006), which is funded by the Portuguese FCT (Fundação para a Ciência e a Tecnologia).

6. References

[1]. Knafla, N. Analysing object relationships to predict page access for prefetching. in Eighth Int. Workshop on Persistent Object Systems: Design, Implementation and Use (POS-8), 1998.
[2]. Bernstein, P., S. Pal, and D. Shutt. Context-based prefetch for implementing objects on relations. in 25th Very Large Data Base Conference (VLDB99), 1999.
[3]. Garbatov, S., J. Cachopo, and J. Pereira. Data Access Pattern Analysis based on Bayesian Updating. in INForum 2009. Lisbon, 2009.
[4]. Garbatov, S., J. Cachopo, and J. Pereira, Data Access Pattern Analysis based on Bayesian Updating. 2009, INESC-ID.
[5]. Garbatov, S., Data Access Patterns Analysis and Prediction for Object-Oriented Applications, in Computer Science. 2009, Technical University of Lisbon, Instituto Superior Técnico: Lisbon.
[6]. Langford, J.W., Logistics: Principles and Applications: McGraw Hill, 1995.
[7]. DoD, U.S., Procedures for Performing a Failure Mode Effects and Criticality Analysis. 1984.
[8]. Kececioglu, D., Reliability Engineering Handbook. Vol. 2. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1991.
[9]. Carey, M., D. Dewitt, and J. Naughton. The OO7 benchmark. in ACM SIGMOD International Conference on Management of Data, 1993.
[10]. Berg, B., Markov Chain Monte Carlo Simulations and Their Statistical Analysis: Hackensack, NJ: World Scientific, 2004.
[11]. Knafla, N. Analysing object relationships to predict page access for prefetching. in Eighth Int. Workshop on Persistent Object Systems: Design, Implementation and Use (POS-8), 1998.

Contacts:
Stoyan Garbatov, MSc and João Cachopo, PhD
Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento,
R. Alves Redol 9, 1000, Lisboa, Portugal
E-mails: stoyan.garbatov@gmail.com
joao.cachopo@inesc-id.pt
Рецензент: доц. д-р инж. А. Антонов, ТУ – Варна



ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ ДЛЯ АВТОМАТИЗАЦИИ РАСПАРАЛЛЕЛИВАНИЯ РЕШЕНИЙ НЕЛИНЕЙНЫХ ЗАДАЧ

Татьяна И. Усова

Резюме: В докладе рассматривается проблема оптимизации работы автоматической системы распараллеливания решения нелинейных уравнений с целью дальнейшей реализации на вычислительном кластере. Предлагается информационная технология распараллеливания решения нелинейных уравнений, основанная на представлении процедуры решения в виде информационного графа.

Information Technology Automation for Parallel Solving of Nonlinear Equations

Tatyana Iv. Usova

Abstract: Problem of automatic system for parallel solving of nonlinear equations performance optimization is considered for the purpose of further implementation on the computing cluster. The Information technology of parallel solving of nonlinear equations bases on a realization of the solving by an information graph.

1. Введение

Получение решения нелинейных задач за приемлемое время является серьезной проблемой во многих областях науки и техники. Методы, используемые на данный момент для решения таких задач, зачастую не удовлетворяют критерию быстродействия; для ускорения решения следует его распараллеливать. Целью данной работы является автоматизация распараллеливания решения нелинейных уравнений (НУ). Предлагаемая ниже информационная технология (ИТ) распараллеливания решения НУ (алгебраических и трансцендентных уравнений) предназначена для автоматизации процесса преобразования алгоритма нахождения корней сложных НУ в параллельную форму, которая в дальнейшем будет реализована на вычислительном кластере.

2. Информационная технология решения сложных задач на вычислительном кластере

Информационная технология для решения задач на кластере включает в себя: подготовку задачи к решению (составление математического описания, выбор метода и алгоритма решения, составление последовательной программы и её отладку), разбиение задачи на достаточно самостоятельные процессы и перевод последовательной программы в параллельную форму, отладку параллельной программы, составление расписания взаимодействия параллельных процессов, балансировку нагрузки процессоров. В случае моделирования должны быть предусмотрены также возможности изменения как математического описания моделируемого объекта (явления), так и его параметров по определенному плану.

Некоторые этапы описанной технологии могут быть автоматизированы (например, с помощью компилятора могут быть распараллелены циклы), но, тем не менее, на плечи пользователя ложится основная тяжесть реализации данной технологии.

Рассмотрим более подробно особенности отладки параллельной программы.

Так как основная причина ошибок в параллельных программах состоит в некорректной работе с разделяемыми ресурсами (возникают взаимоблокировки процессов), то основной задачей является определение условий их возникновения. Сложность в том, что недетерминированность поведения параллельной программы не позволяет выявить предшествовавшее ошибке поведение программы, т.е. причину ошибки. Необходимо, чтобы отладчик, анализирующий поведение параллельной программы (с целью улучшения производительности), обладал инструментами наблюдения (мониторинга) и анализа (визуализации). Монитор желательнее программный, как сочетание простоты и высокого уровня гибкости и применимости, но здесь существует трудноразрешимая проблема кроссплатформенности. А визуализатор должен оперировать следующими характеристиками содержимого памяти компьютера: областью данных программы (переменные и константы), адресным пространством (буферы, стеки, код программы), состоянием процессора, вводом-выводом. Установить порядок наступления событий в распределенной системе можно с помощью пространственно-временной диаграммы Л. Лемпорта [1].

В соответствии с законом Амдала [2] ускорение параллельной вычислительной системы (ПВС) зависит от потенциального параллелизма задачи и числа процессоров n. Однако закон Амдала не отражает потерь времени на межпроцессорный обмен сообщениями, которые зависят от свойств коммуникационной среды, и на передачу служебной информации, определяемой программным обеспечением. Параметр, учитывающий эти потери, называется коэффициентом деградации; он определяет сетевой закон Амдала [2]. Для ускорения вычислений следует воздействовать и на алгоритмическую, и на техническую составляющие коэффициента деградации. Для программирования кластеров применяются библиотеки функций, построенные по модели обмена сообщениями, – MPI [3].

3. Автоматизация распараллеливания решения нелинейных уравнений

ИТ распараллеливания решения нелинейных уравнений основана на разбиении НУ на элементарные структурные единицы в соответствии с [4]. В зависимости от сложности первоначальной структуры уравнения разбиение может происходить рекурсивно в определённое число шагов до тех пор, пока полученные элементы или не будут содержать знаки операций и числа, или сами не будут представлять те виды структурных единиц, которые были приняты за элементарные.

ИТ распараллеливания НУ включает в себя следующие этапы.

Этап 1. Выделение элементарных структур. На входе системы мы получаем сложное алгебраическое или трансцендентное уравнение. На выходе генерируется совокупность структурных единиц, из которых состоит НУ. На данном этапе процесс выделения структур может корректироваться экспертом.

Этап 2. Полученная совокупность структурных единиц представляется в виде информационного графа. В качестве узлов графа могут выступать только числа и знаки операций. Если на этапе разбиения были использованы элементарные структурные единицы, определенные в [4], то здесь они представляются уже не одним узлом, а подграфом.

Этап 3. Строится информационный граф алгоритма метода решения нелинейного уравнения, определяемого экспертом. Построенные на этапе 2 информационные графы уравнения и его производных объединяются с основным графом – графом алгоритма нахождения корней уравнения.

Этап 4. Преобразование информационного графа решения уравнения в кадровую ярусно-параллельную форму [5].

Этап 5. Ярусно-параллельная форма информационного графа преобразуется в структурно-процедурную форму с учетом того, что кадры располагаются по уровням, а каждый кадр рассматривается в качестве фундаментального оператора [5]. В конце этого этапа создается структурно-процедурная программа, которая может быть реализована на многопроцессорной системе.

Исходными данными для получения структурно-процедурной программы решения нелинейного уравнения являются:

1. сложное алгебраическое или трансцендентное уравнение;
2. границы отрезка локализации корня;
3. заданная точность решения уравнения.

На первых трех этапах распараллеливания решения важную роль играет экспертная система. В процессе работы экспертной системы происходит построение информационного графа уравнения и, в зависимости от выбранного алгоритма решения, информационных графов производных. На основании построенных графов уравнения, производных и алгоритма нахождения корней строится общий граф решения.

На этапе разбиения нелинейного уравнения по структуре используется база знаний, содержащая в себе набор уже известных элементарных структурных единиц. В процедурах структурного разбиения и построения информационных графов принимает участие эксперт, имея возможность внести свои коррективы в работу системы.

Нелинейное уравнение должно быть представлено в виде графа определенного вида, так как в дальнейшем он будет объединяться с информационным графом алгоритма решения уравнения данного типа. Система автоматически выбирает один из алгоритмов итерационного нахождения корней. В этом случае часто приходится множество раз вычислять значение функции при различных значениях аргумента. Кроме того, большинство итерационных методов решения использует производные первого порядка и выше. Соответственно, необходимо получение аналитического вида этих производных, а также информационных графов, соответствующих им. Если получение аналитического выражения для производных невозможно, система может использовать, алгоритмы численных методов нахождения производных функций, хранящиеся в ее БЗ. Именно от эксперта будет зависеть, какой алгоритм решения, и какая точность вычисления производных (если в этом есть необходимость) будут выбраны для заданного уравнения.

После окончания работы экспертной системы полученный граф решения автоматически преобразуется в кадровую ярусно-параллельную форму.

Распараллеливание данного решения получило развитие с использованием структурно-процедурной реализации вычислений для систем с массовым параллелизмом [6], когда исходный ИГ преобразовывается в кадровую форму, где каждый кадр представляет собой программно-неделимый участок кода (структурная компонента), а набор кадров является процедурной компонентой, последовательно выполняющейся на МВС. Каждый кадр внутри может содержать множество операционных вершин, которые являются независимыми друг от друга и выполняются параллельно. С учетом того, что каждый кадр может содержать произвольное число операционных вершин (от одной до нескольких сотен для сложных задач), неизвестно, сколько именно процессоров будет задействовано.

Решение такой проблемы предложено в [7], где ИГ представляется в ярусно-конвейерной форме, для более четкого определения предшествующих и последующих вершин. Каждой вершине ставится в соответствие свой процессор, с учетом построения к нему наименьшей трассы, с использованием карты волновых фронтов для определения возможности участия данного процессора в формировании маршрута. Если для каких-то узлов МВС трассу создать не удается, происходит откат и новое формирование трассы на основе других узлов. Этот алгоритм может быть использован для различных коммутационных структур и для графов содержащих как обратные связи, так и циклические участки. Однако, его использование целесообразно для

масштабных задач, когда время трассировки будет несоизмеримо мало по сравнению со временем вычислений. В нашем случае ситуация иная, и время, которое будет потрачено на трассировку значительно ухудшит временные показатели вычислений самого НУ. В связи с этим предлагается использовать динамические структуры для задания количества узлов кластера, предполагая, что время пересылки данных между ними одинаково. В этом случае можно использовать структурно-процедурную реализацию вычислений из [6] для кластера, когда процедурная компонента, переходя от кадра к кадру, будет динамически изменять число используемых процессоров, что приведет к более разумному использованию ресурсов.

4. Заключение

Таким образом, предложена информационная технология решения сложных задач на вычислительном кластере, причем более подробно рассмотрены особенности отладки параллельных программ. Для оценки ускорения решения на кластере относительно последовательной программы предлагается использовать сетевой закон Амдала.

Разработанная методика автоматизации построения ИГ с использованием экспертной системы позволяет распараллеливать различные виды задач из класса нелинейных за счет инвариантности модели решений относительно вида уравнений и дает возможность объединить алгоритм структурного анализа нелинейного уравнения и структурно-процедурную реализацию на многопроцессорной системе с программируемой архитектурой.

Литература

[1]. Lamport L., Time, Clocks, and the Ordering of Events in a Distributed System, Communications of the ACM, 1978

[2]. Воеводин В.В., Вл.В. Воеводин, Параллельные вычисления, СПБ, БХВ, 2004

[3]. Евсеев И., MPI: Вводный курс, http://www.ssd.sscc.ru/old/kraeva/MPI.html

[4]. Паулин О.Н., Т.И. Усова, Метод распараллеливания нелинейных задач, Проблемы программирования, 2010

[5]. Каляев И.А., И.И. Левин, Е.А. Семерников, В.И. Шмойлов, Реконфигурируемые мультиконвейерные вычислительные структуры, Ростов-на-Дону, ЮНЦ РАН, 2008

[6]. Левин И.И., Ресурсонезависимое параллельное программирование, Таганрог, ТРТУ, 2002

[7]. Левин И.И., Реализация информационных графов в архитектуре многопроцессорных систем, Таганрог, ТРТУ, 2002.

Для контактов:
асс. Татьяна Ивановна Усова
кафедра Системного программного обеспечения
Национальный политехнический университет– Одесса
E-mail: usovati@rambler.ru

Рецензент: доц. д-р инж. Н. Николов, ТУ – Варна

A Three-Dimensional Computational Fluid Dynamics Model of a Rocket

Latchezar I. Georgiev

Abstract: This paper proposes a 3D computational fluid dynamics (CFD) model of a rocket to visualise the oblique shock-wave cone at a certain Mach number. Its angle is then compared to the angle of a real such cone on a single still film frame to verify the accuracy of the goniometric algorithm for rocket velocity computation from this frame. The Mach number is adjusted iteratively to minimise the difference. This model of the *Saturn V* rocket at *S-IC* staging time verifies the Mach number computation method defined earlier.

Триизмерен аеродинамичен изчислителен модел на ракета

Лъчезар Ил. Георгиев

Резюме: В статията се предлага триизмерен аеродинамичен изчислителен модел на ракета с цел изобразяване на конуса на ко́сата ударна вълна при дадено число на Мах. Неговият ъгъл се сравнява с ъгъла на реалния конус от филмов стоп-кадър за проверка на точността на гониометричния алгоритъм за изчисляване на скоростта на ракетата по този кадър. Числото на Мах се коригира итеративно, за да се получи възможно най-малка разлика. Този модел на ракетата „Сатурн-5“ при отделяне на степен „*S-IC*“ проверява метода за изчисляване на числото ѝ на Мах, дефиниран по-рано.

1. Introduction

Above a certain supersonic Mach number, a conical shock wave is attached to the aircraft cone's apex. The wave's cone angle is often used to determine the Mach number [1] (in wind tunnels to test airfoils, engines, etc.). Pokrovsky in 2007–2010 and Popov in 2010 did this to a rocket in open air [4], analysing NASA's *Apollo 11* launch clip [5] and its still frame [6] by this and three other methods. An author's paper defined a rocket velocity computation formula and algorithm to get a more accurate rocket speed value [7]. This paper aims at verifying the formulae, the algorithm, and the Mach number obtained through it by modelling the supersonic airflow past the rocket at this Mach number and comparing the angle of the so obtained oblique shock-wave to the measured one.

2. Model creation steps

The most widely used CFD software, *Fluent*®, was selected as simulation "workhorse". The following objects were created in its geometric modelling and mesh-generation software, *Gambit*®:

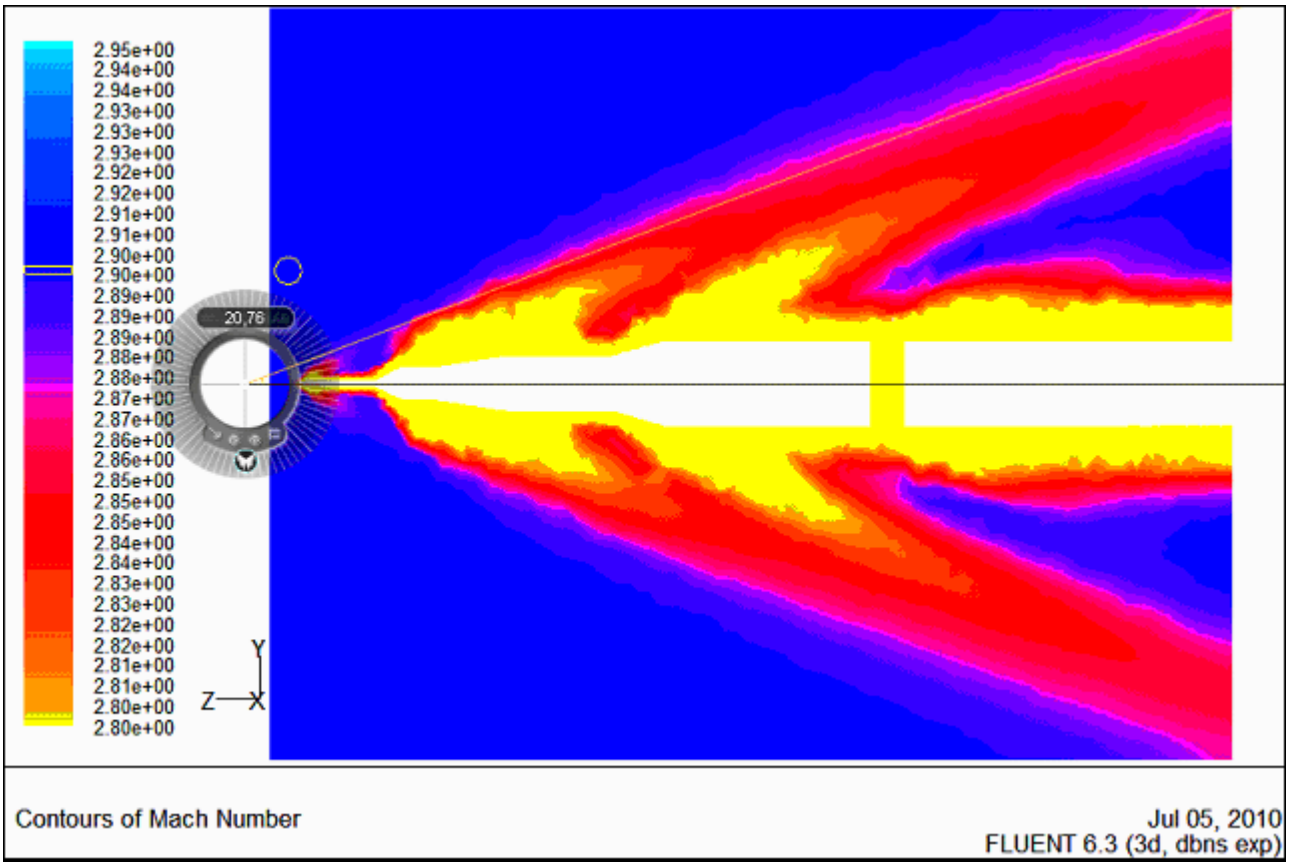
1. A real size (±1 *mm*), simplified 3D drawing of the *Saturn V* rocket at separation time.
2. A parallelepiped "brick" volume for the airflow around the rocket and connected to it.
3. A triangular element surface mesh for the launch-escape motor with a spacing of 0,2 *m*.
4. A triangular element surface mesh for the rest of the rocket with a spacing of 0,47 *m*.
5. A tetrahedral element volume mesh for the modelled airflow with a spacing of 2 *m*.
6. A surface group named "rocket" uniting all the rocket surfaces.
7. A surface group named "farfield" uniting all the "brick" surfaces.
8. A boundary zone of type *WALL* named "rocket" – from the "rocket" group.
9. A boundary zone of type *PRESSURE_FAR_FIELD* named "farfield" – from that group.

A mesh file was exported to *Fluent* and imported there. The following operations were done there:

A. Scaling of the imported grid 1:1000 to be in metres (was defined in millimetres in *Gambit*).
B. Reordering the grid (both the domain and zones) to increase the memory access efficiency.

C. Selecting the explicit density-based solver as the flow is supersonic and thus compressible.

Fig. 1: Cross-Z-axis section of 3D *Saturn V* rocket shock wave cone model at Mach 2.9.
Solver acronyms: "dbns" = Density-Based Navier-Stokes, "exp" = explicit, "imp" = implicit.



D. Selecting the Green-Gauss node-based gradient option for better accuracy.
E. Setting the viscous model to *inviscid*.
F. Enabling the energy equation for the model.
G. Setting the fluid material to *air* and density to *ideal gas*.
H. Setting operating pressure condition to 0 to make absolute pressure equal to gauge pressure
I. Setting the "farfield" boundary condition to 270.65 *K*, 101.325 *Pa* and Mach 2.9 [7].
J. Setting the *X*- and *Y*-components of flow direction to 0, and the *Z*-component to −1.
K. Setting the minimum absolute pressure solution limit to 0.1 *Pa*.
L. Setting the following flow equation solution controls: flow discretisation = *first order upwind*, Courant number = 2, residual smoothing iterations = 2, smoothing factor = 0.5.
M. Computing the solution initialisation from the "farfield" zone.
N. Initialisation of the solution.
O. Enabling the *plot* option of the residual monitors.
P. Doing 1000 iterations. The solution converged at iteration 821 in ≈80 *min* (*Core Duo/2GB*)

The computed Mach 2.863 [7] resulted in a slightly greater angle. A second iteration at Mach 2.9 gave the 20.76° angle measured in [7] (fig. 1). So, the computed Mach number error is just 1%.

4. Remarks

Note that the maximal measurement error of 7.76% in [7] is valid here too, as the measured oblique shock wave cone angle is the basis for comparison to the simulated shock wave cone angle.

The notion that the phenomenon seen on NASA photo [6] is really a conical shock wave (and is therefore to be modelled as such) received confirmation by aerospace / aeronautics engineers [8]. And the modelled flow (fig. 1) is similar to the wind tunnel shadowgraph of a real Saturn V model [9] whose shock cone angle of 28° at Mach 1.93 correlates well with the 20.76° angle at Mach 2.9. A 7-zip archive of the source database file for Gambit and case and data files for Fluent of the model is freely available for download and examination [10]. It is hereby declared “public domain”.

4. Rocket velocity

Despite the correct Mach number, the rocket velocity calculation in [4] and [7] is wrong, as the retrorocket exhaust gases surrounding the rocket are much hotter than the counter air flow, which is not taken into account there. It is difficult to calculate the exact temperature of these gases (and thus the local speed of sound), but the rocket velocity can accurately be computed via the well-known Tsiolkovsky’s formula substituting the mass ratio in it from the specific impulse formula [3]

$$I_{sp0} = \frac{F_0}{\frac{M_1 - M_2}{t_b}} \tag{1}$$

where F0 is the sea level thrust, Isp0 is the specific impulse at sea level, tb is the burn time, M1 is the start mass, and M2 is the mass at time tb. As F0 can be expressed as the product of the thrust-to-mass ratio (a.k.a. G-force) Gf and the start mass M1, formula (1) can be transformed as follows:

$$\frac{M_1 - M_2}{t_b} = \frac{F_0}{I_{sp0}}$$

$$\frac{Gf \cdot M_1}{I_{sp0}} = \frac{M_1 - M_2}{t_b}$$

$$\frac{Gf \cdot M_1 \cdot t_b}{I_{sp0}} = M_1 - M_2$$

$$M_2 = M_1 - \frac{Gf \cdot M_1 \cdot t_b}{I_{sp0}}$$

$$M_2 = M_1 \cdot \left(1 - \frac{Gf \cdot t_b}{I_{sp0}}\right)$$

$$\frac{M_2}{M_1} = 1 - \frac{Gf \cdot t_b}{I_{sp0}}$$

$$\frac{M_1}{M_2} = \frac{1}{1 - \frac{Gf \cdot t_b}{I_{sp0}}} \tag{2}$$

Substituting the mass ratio (2) in the Tsiolkovsky's formula [11] below, we get (3):

$$V_{ch} = g_0 \cdot I_{sp} \cdot \ln\left(\frac{M_1}{M_2}\right)$$

$$V_{ch} = g_0 \cdot I_{sp} \cdot \ln\left(\frac{1}{1 - \frac{Gf \cdot t_b}{I_{sp0}}}\right) \quad (3)$$

where Vch is the characteristic velocity, g0 is the Earth's gravity, and Isp is the velocity-integrated mean specific impulse (slightly less than the specific impulse in vacuum). Substituting in (3) the known g0 = 9.80665 m/s2 [2], Gf = 1.195 [12], tb = 161.63 − 0.3 − (161.63 − 135.2) / 5 = 156.044 s [14] (adjusted for all the five engines working all the time), Isp = 304 s, and Isp0 = 265 s [13], we get:

$$V_{ch} = 9.80665 \times 304 \times \ln\left(\frac{1}{1 - \frac{1.195 \times 156.044}{265}}\right) = 3626 \text{ m/s} \quad (4)$$

Subtracting the gravitational (1180) and aerodynamic (47) losses computed with [15], we get 3626 − 1180 − 47 = 2399 m/s. This is just 0.24% higher than the declared value of 2393.3 m/s [14]. (Interested readers can refer to Braeunig's refined and detailed SA-506 / Apollo 11 simulation [16].) Note that no value substituted above could differ noticeably. E.g. if Gf differs, the time the rocket clears the umbilical tower would differ, which will be noticed; same for tb. And if any other but the central engine is cut off, flight trajectory control by gimballing the working engines could not work. Single-sequence footage [17] confirms the Mach 1, central engine cut-off, and staging times in [14].

5. Conclusion

A 3D computational fluid dynamics model of the Saturn V rocket has been created. The goniometric algorithm for rocket Mach number computation defined by the author in [7] has been verified by it and found to be exact within 1%. This simulation gave a more exact real Mach number of 2.9. However, the temperature of the retrorocket exhaust gases surrounding the rocket at the moment the photo [6] was taken was much higher, and so was the rocket speed. So, Boeing's value of ≈2.4 km/s [14] is correct, whereas the low velocity values in [4] by Pokrovsky and Popov and in [7] by the author are wrong, which this paper verifies via Tsiolkovsky's and specific impulse formulae.

6. Literature

[1]. Marks, L., Marks' Standard book for mechanical engineers, McGraw-Hill, New York, 2007, p. 11-74
[2]. Ibid, p. 1-25
[3]. Ibid, pp. 11-90–91
[4]. Покровский, С., А. Попов, Расследования: Луна, «Сверхновый проект», Москва, 2010 г.
[5]. Dismukes, K., J. Petty, A view of the first stage separation of Apollo 11, NASA, Hampton, 2002
[6]. Dismukes, K., A. Kauderer, Apollo imagery – S69-39957, NASA, Hampton, 2009
[7]. Georgiev, L., Goniometric rocket velocity computation algorithm, CST, TU–Varna, 1/2009, pp. 26–31
[8]. Murphy, D., What supersonic aerodynamics phenomenon is seen?, Tecumseh Group, Southfield, 2010
[9]. Andrews, C., D. Carlson, Shadowgraph study of upper stage flow fields, NASA, Huntsville, 1965, p. 21

[10]. Georgiev, L., Gambit/Fluent case/date file and image archive, CSE Department, TU–Varna, 2010
[11]. Benson, T., Ideal rocket equation, NASA, Cleveland, 2002
[12]. McKay, G., Saturn V launch vehicle flight evaluation report–AS-506, NASA, Huntsville, 1969, p. 11-6
[13]. Ibid, p. 5-4
[14]. Krausse, S., Apollo/Saturn-V postflight trajectory–AS-506, Boeing Co., Huntsville, 1969, pp. B-26–30
[15]. Pustynski, V., L. Georgiev, AS-506 (S-IC-6) ascent simulation, Institute of Physics, TU–Tallinn, 2010
[16]. Braeunig, R., Saturn V launch simulation, Rocket and Space Technology, Dayton, 2010
[17]. Apollo 11 tracking camera, Google Inc., San Bruno, 2007

For contacts:
Ass.-Prof. Eng. Lâtchezar I. Georgiev
Computer Science and Engineering Dpt.
Technical University of Varna
E-mail: lucho@gawab.com
Рецензент: доц. д-р инж. П. Антонов, ТУ – Варна

ОТНОВО ЗА ТРЕТАТА СТЕПЕН НА ВИСШЕТО ОБРАЗОВАНИЕ – ДОКТОРАНТУРАТА

Ангел С. Смрикаров

Резюме: В статията е дадена една примерна методика за работа върху дисертацията, в която са отбелязани основните моменти на една докторантура – формулиране на проблем и тема, анализ на известните решения, формулиране на цел и задачи, теоретично изследване, реализация, експериментално изследване, формулиране на приносите. В заключението авторът излага своето виждане за задълженията на научния ръководител на докторанта.

Again About the Third Education Level – Doctoral Degree

Angel S. Smrikarov

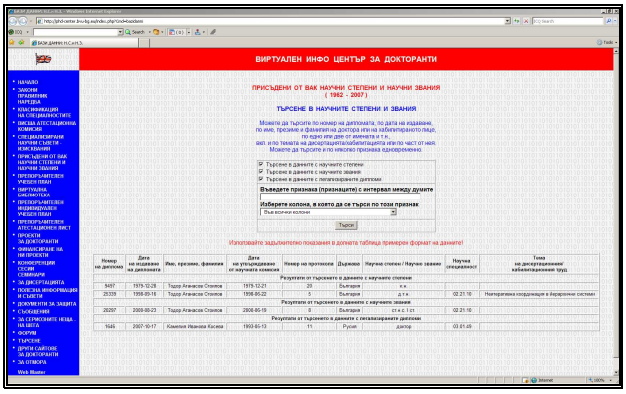
Abstract: This paper describes a sample procedure how to work on a doctoral dissertation. It marks the main stages in a doctoral study – formulation of a problem and title, analysis of existing solutions, formulation of an aim and tasks, theoretical investigation, implementation, experimental investigation, formulation of contributions. In the conclusion the author gives his view about the obligation of the doctoral student's supervisor.

Бях решил да обоснова необходимостта от написването на този материала с ниското качество на докторантурата, както и с това, че почти всеки, който успее да събере 150-200 стр. между две черни корици и направи 10-15 плаката, получава заветната докторска степен, но прецених, че нито докторантите, нито техните ръководители, нито пък все още (доскоро) действащите научни съвети носят изцяло вината за това. Съобразих, че и аз съм един от тези, които, имат вина за това положение на нещата и затова реших да смекча тона и да се задоволя с даване на някои съвети на тези, за които качеството като понятие все още не е загубило своя смисъл. Но още в самото начало бих искал да подчертая, че тези съвети са валидни основно за дисертации в областта на техническите науки.

Наскоро, на един докторантски семинар, попитах, с какво започва една дисертация? Получиха се много и смислени отговори, но нито един не беше напълно верен. А отговорът се съдържа в самото определение за дисертация, според което това е „Итеративен процес на решаване на актуален **проблем** от теорията или практиката с прилагане на научен подход и с използване на научни методи и средства". Ето как, например, може да звучи един проблем: "По данни на наши и чужди автори около 60% от времето, необходимо за създаване на нова колесна машина или за нейното модифициране се изразходва за изпитване и изследване. Значителна част от това време се отделя за определяне на динамичните свойства на машината. Една от основните причини за това е липсата на функционално пълна система за автоматизация на инженерния труд при изследването на колесни машини в процес на ускоряване и спиране."

След идентифицирането на проблема се формулира работното заглавие (темата) на дисертацията. Желателно е то да бъде кратко, но точно и ясно, като е препоръчително да започва с отглаголно съществително, като напр. „Изследване и създаване на ……..……"

или „Създаване и изследване на ……..……..……”, в зависимост от реда на етапите, акцента и т.н.



Фигура 1. Уеб базиран регистър на присъдените научни степени и научни звания

След като темата бъде формулирана, е желателно да се провери, дали дисертация със същото или подобно заглавие не е вече разработена и защитена. За целта може да се използва базата от данни с присъдени от Висшата атестационна комисия научни степени и научни звания, която е интегрирана във Виртуалния инфо-център за докторанти и е общодостъпна, и в която търсенето на информация може да става по различни признаци, в т.ч. и по ключови думи (Фиг. 1).

http://phd-center.bvu-bg.eu/index.php?Cmd=bazidanni

В процеса на докторантурата темата може да претърпи известна промяна с цел да отрази по-точно съдържанието на дисертацията. Това, съгласно Наредбата за държавните изисквания за приемане и обучение на докторантите, може да стане не по-късно от 3 месеца преди датата, определена за представяне на дисертационния труд за защита.

Много важно е, още в самото начало, да бъде съставена методиката за работа върху дисертацията. Тази методика задължително трябва да включва използването на научни методи и средства. Например, сигурен признак за научен подход към решаването на проблема е използването на метода на моделирането, последвано от теоретичното изследване на съставения модел. Една такава методика е показана на фиг. 2.

Важно е докторантът да знае, как трябва да бъде оформен крайният резултат от неговия труд. Ето защо по-долу е дадена една примерна структура и съдържание на дисертация.

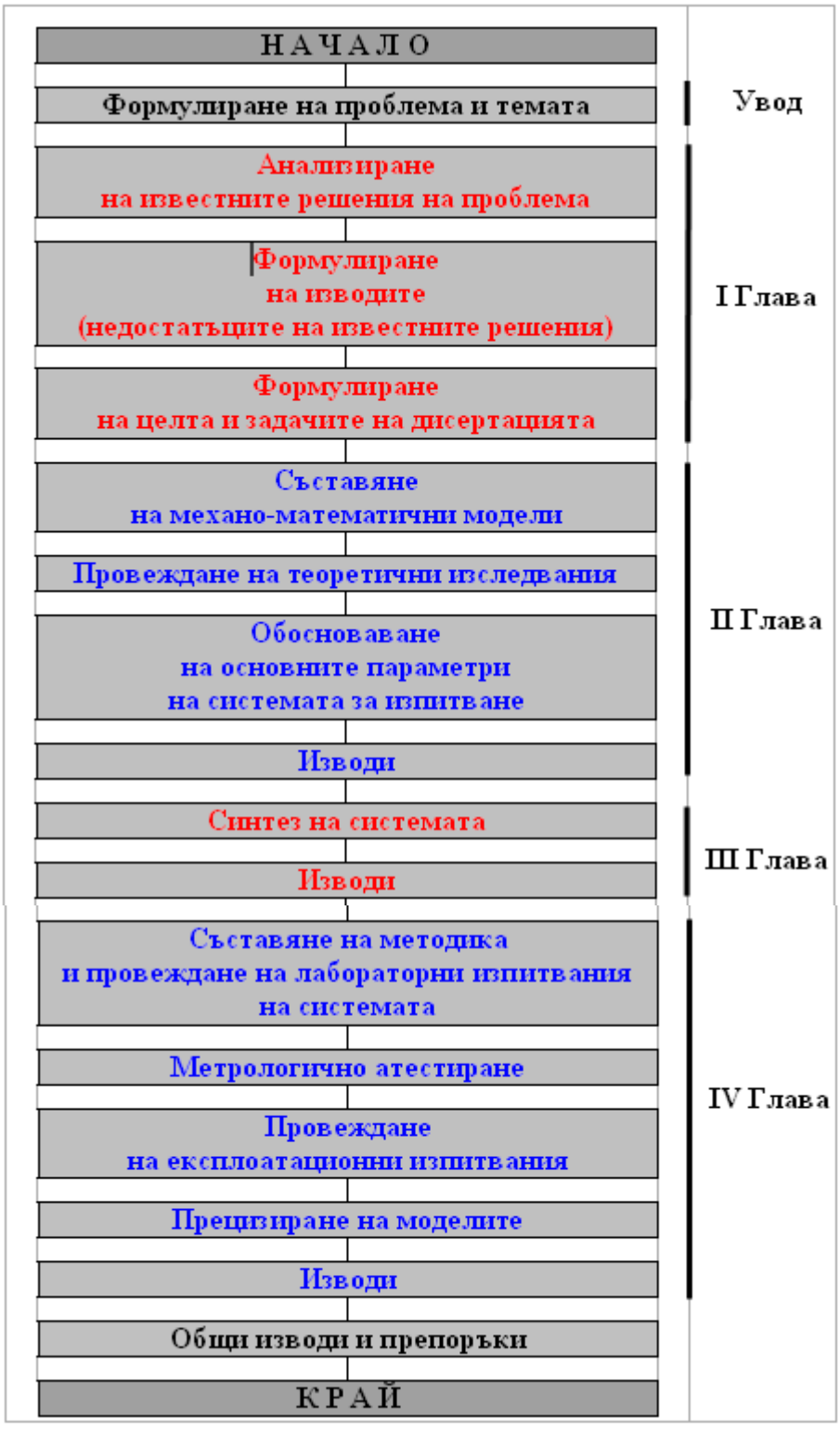
Списък на използваните съкращения
Списък на фигурите
Списък на таблиците
Анотация (кратко съдържание на дисертацията, 1 стр.)
Увод (формулировка на проблема и обосновка на неговата актуалност, 1-2 стр.)

I Глава (около 30 стр.)
Анализ на състоянието на проблема
(Анализ на предшестващите резултати от научните изследвания по проблема)

1.1.

1.2.

…..



Фигура 2. ***Примерна методика на работа върху дисертация на тема “Изследване и създаване на система за изпитване на транспортни машини”***

Изводи (предимства и недостатъци на известните решения на проблема; нерешени задачи)

Цел и задачи на докторантурата (целта е една, а задачите 4-6) Следващите глави трябва да съдържат решенията на формулираните задачи, което трябва да бъде направено с използване на научни методи и средства и да води до достигане на поставената цел, т.е. до отстраняването на недостатъците на известните решения. Формулировката на целта обикновено включва в себе си темата на дисертацията, като например:
Тема: Създаване и изследване на система за ….
Цел на дисертационния труд е създаването и изследването на функционално пълна система за …… , която ……)

II Глава (около 30-40 стр.)
Теоретични изследвания
2.1.
2.2.
…..

Тази глава на дисертацията е най-важна. В нея докторантът трябва, например, да формулира критерий за оптималност, да предложи методика за теоретично изследване и модел, да представи и анализира резултатите от изследването. Ако предмет на дисертацията е създаването на някаква система, то резултатите от теоретичните изследвания биха могли да бъдат, например, стойностите на параметрите на системата, при които тя ще функционира оптимално от гледна точка на формулирания критерий.
Резултати и изводи

III Глава (около 30 стр.)
Практическо решаване на проблема (с научни методи и средства)
3.1.
3.2.
…..
Резултати и изводи

IV Глава (около 30 стр.)
Експериментални изследвания (по научно обосновани методики)
4.1.
4.2.
…..
Резултати и изводи
Забележка: Резултатите от експерименталните изследвания в идеалния случай трябва да съвпадат или да са много близки до тези от теоретичните. На практика могат да се получат различия, които най-често се дължат на несъвършенството на модела, използван при теоретичните изследвания. В такъв случай моделът се усъвършенства и т.н.
Общи изводи (обобщение на частните изводи след всяка глава)
Предложения за използване на резултатите и виждания за насоките на по-нататъшната работа
Заключение (основните приноси - предложени, разработени, създадени нови или модифицирани методи, методики, алгоритми, модели, устройства, технически и/или програмни системи и др. с доказана полезност за практиката; от приносите трябва да се разбира, че поставените задачи са решени и то - с използване на научни методи и средства и че целта на докторантурата е постигната)
Използвана литература (не по-малко от 110 заглавия - монографии, книги, учебници, научни статии и доклади, фирмена литература и сайтове, повечето издадени през последните 10-15 години, от които над 50 % на латиница; използваната литература трябва да бъде

тематично максимално близка до дисертационния труд и да дава пълна и точна картина за състоянието на проблема; всички заглавия без изключение трябва да бъдат цитирани в дисертацията)

Приложения (резултати от теоретични и експериментални изследвания, проведени както от автора, така и от други лица и организации, които са необходимо допълнение на дисертацията, но не са намерили място в основната й част, например, поради ограниченията в обема)

Публикации - всички **основни резултати** на докторанта трябва да бъдат публикувани в научно-технически списания и в сборници на научно-технически конференции. Броят на публикациите трябва да бъде минимум 5-6, от които:

- една публикация в чуждестранно списание или в сборник на международна конференция - желателно;
- минимум една самостоятелна публикация;
- минимум две публикации в съавторство с научния ръководител, като е желателно докторантът да бъде на първо място в списъка на съавторите.

Всяка публикация трябва да съдържа примерно:

- увод - проблем, недостатъци на известните решения, задача за решаване;
- изложение - описание на създадените с цел решаване на задачата методи и средства; основни резултати; анализ на резултатите;
- заключение - основни изводи и препоръки;
- литература – мин. 4-6 заглавия, от които 1-2 на латиница.

Внедрявания на резултатите (2-3 - в учебния процес, във фирми и др. - удостоверяват се със служебни бележки, в които трябва да има и информация за ефекта от внедряването)

Забележка: Горният план касае формата на дисертацията и е примерен. Във всеки конкретен случай трябва да се изготвя конкретен план в зависимост от конкретната тема, от изискванията, приети в съответния университет или научна област и т.н. Без да се подценява формата, не бива да се забравя, че основното е съдържанието на дисертацията, чрез което докторантът доказва, **че е овладял методологията на научното изследване и че е в състояние да я прилага за решаване на важни за практиката задачи.**

ВАЖНО: Изводите след първа глава, задачите и приносите трябва да са логически свързани. От една страна, на всеки извод, в който се изтъква някакъв недостатък на известните решения на проблема, трябва да съответства задача, която трябва да е така формулирана, че нейното решение да води до отстраняване на недостатъка. От друга страна, успешното решаване на поставената задача е принос на докторанта, който може да бъде научен, научно-приложен или приложен. Броят и най-вече качеството на тези приноси е от съществено значение за крайното решение на съответното научно жури. За да се постигне съответствие между изводите, задачите и приносите може да се използва таблица като долната, която може да бъде наречена **“Дисертационна матрица”** поради наличието на логически връзки не само по хоризонтала, за което става дума по-горе, а също и по вертикала, в смисъл че изводите, респ. задачите и приносите трябва да следват в строго определена логическа последователност.

No	Изводи (след първа глава)	Задачи	Приноси
1.	.....	.....	.....
2.	.....	.....	.....

3.	Известните методи не дават възможност за измерване на ..... с достатъчна точност.	Да се създаде метод, който да позволява измерването на …..... с относителна грешка не по-голяма от 1%.	Предложен е метод, който дава възможност за измерване на …..... с относителна грешка по-малка от 0,8%, с което от една страна се удовлетворяват изискванията на съответните стандарти, а от друга се повишава научната и практическата стойност на резултатите от измерването. Методът е патентован.
4.	.....	.....	.....
5.	.....	.....	.....
6.	.....	.....	.....

Таблица 1. Дисертационна матрица

Забележка: Срещу някой от изводите може и да няма задача, респ. принос, а срещу други може да има и повече от една задача, респ. принос.

В Германия наричат научните ръководители на докторантите "**Doktorvater**", буквалният превод на което е – **баща на докторанта**. Споделяйки напълно това становище за ролята на научния ръководител, в заключение бих искал да предложа за дискусия едно виждане за неговите задължения, изпълнението на които действително би превърнало всеки ръководител в научен баща на своя докторант.

1. Да селектира и насочва към докторантура млади хора с действителни способности и желание за научна и преподавателска дейност.
2. Да формулира актуален за практиката проблем, който може да бъде решен с научни методи и средства, както и дисертабилна тема.
3. Да съдейства при подготовката за **конкурсния изпит** (конспект, литература, консултации, комисия) и при **зачисляването в докторантура**.
4. Да създава нормални условия за работа на докторанта:
   - осигуряване на благоприятен микроклимат;
   - определяне на подходящо работно място;
   - предоставяне на компютърна техника;
   - осигуряване на необходимото лабораторно оборудване;
   - включване в проекти с подходяща тематика, участието в които да носи на докторанта конкретни морални и материални ползи;
   - включване в семинари;
   - изпращане на специализации;
   - подпомагане подготовката и изнасянето на доклади на научни конференции, в т.ч. и международни – командировъчни, такси правоучастие;
   - подпомагане подготовката и публикуването на статии в научни списания, в т.ч. и международни – такси;
   - съдействие за популяризиране публикациите на докторанта с оглед тяхното евентуално цитиране от други автори;

- провеждане на редовни срещи и обсъждане на постигнатите резултати и предстоящите задачи.

5. Да съдейства при съставянето и изпълнението на **индивидуалния план** на докторанта.
6. Да стимулира участието в курсовете за обща подготовка на докторантите.
7. Да спомага при формулирането на изводите от обзора (недостатъците на известните решения на проблема), целта и задачите на дисертацията.
8. Да насочва към използването на научни методи и средства при решаването на задачите.
9. Да насърчава самостоятелната работа и творческото мислене на докторанта.
10. Да съдейства, при необходимост, за провеждане на консултации с други изявени специалисти в научната област на докторантурата.
11. Да съдейства при съставянето на ежегодните отчети и атестации на докторанта.
12. Да съдейства при подготовката и провеждането на кандидатския минимум (конспект, литература, консултации, комисия).
13. Да спомага при цялостното структуриране на дисертацията.
14. След решаването на задачите на дисертацията и събиране на достатъчен по обем материал да насочи докторанта към приключване на работата и написване на дисертацията.
15. Да съдейства при формулирането на общите изводи и приносите.
16. Да съдейства за своевременната защита на създадената в процеса на докторантурата интелектуална собственост.
17. Да спомага за внедряването на получените резултати в практиката.
18. Да спомага за популяризирането на резултатите на докторанта чрез участие в изложби и др.
19. При получаване на значими резултати да предлага докторанта за награждаване.
20. Да съдейства за организирането на **вътрешната защита** (научно звено, плакати, презентация, експозе, компетентни рецензенти).
21. Да съдейства при подготовката за **официалната защита** (срещи с рецензенти, отзиви и др.).
22. Да съдейства за привличане на защитилия докторант към преподавателска работа по подходяща дисциплина.
23. Да го поощрява да продължава да се занимава с наука, за да „трупа", в частност, актив за **повишаване в академична длъжност**, но преди всичко – за да бъде полезен на студентите, на катедрата и на себе си.

Ще се съглася с всеки, който би възразил, в смисъл, че тези изисквания са много и тежки за изпълнение. Но кой е казал, че задълженията на един истински научен баща трябва да са малко и леки?

За контакти:
доц. д-р инж. Ангел Сотиров Смрикаров,
зам. ректор НКР
Русенски университет "Ангел Кънчев"
E-mail: ASmrikarov@acs.uni-ruse.bg

ПРОЕКТ ПО ОПЕРАТИВНА ПРОГРАМА „РАЗВИТИЕ НА ЧОВЕШКИТЕ РЕСУРСИ”

Георги С. Тодоров

Продължава изпълнението на проект “Подкрепа на творческото развитие на докторанти, пост-докторанти и млади учени в областта на компютърните науки”, финансиран от Европейски социален фонд, Оперативна програма “Развитие на човешките ресурси” и Схема за предоставяне на безвъзмездна финансова помощ: “Подкрепа за развитието на докторанти, пост-докторанти, специализанти и млади учени”.

Проектът се изпълнява от консорциум, включващ висши училища и институти на БАН (Русенски университет "Ангел Кънчев”, Софийски универитет “Климент Охридски”, Пловдивски университет “Паисий Хилендарски”, Технически университет - Варна, Технически университет - Габрово, Технически университет - София и Институт по информационни и комуникационни технологии - БАН). Проектът се координира от Великотърновския университет “Св.Св. Кирил и Методий”.

Целева група са 51 млади учени, докторанти и пост-докторанти от организациите-партньори. Значителна част от тях са членове на Съюза по автоматика и информатика.

Разработката стартира през м. септември 2009 г. и основните резултати, които са получени досега се свеждат до:

- Провеждане на две обучителни интензивни програми през м. януари и м. септември 2010 г. По време на обучението крайните бенефициенти изслушаха цикъл от лекции на изтъкнати специалисти от областта на компютърните науки и информатиката. С това те обогатиха своите знания и се запознаха с нови методи и средства за провеждане на научните изследвания по своите теми. Като лектори участваха проф. Пл. Боровска, проф. Н. Синягина, проф. Т. Стоилов, Проф. Ст. Капралов, доц. М. Тодорова, доц. П. Мануилов, доц. О. Асенов, доц. Кр. Стоилова, доц. О. Георгиева;
- С оглед подпомагане на научните изследвания всички крайни бенефициенти бяха абонирани за издания на международните професионални организации IEEE и ACM;
- Във всяка една от партньорските организации се изгради Лаборатория на младия учен, оборудвана със съвременна компютърна техника и друга специализирана апаратура. По заявки на отделните партньори беше закупено и инсталирано специализирано програмно осигуряване;
- Подготвя се провеждането на Кръгла маса с участиетио на крайните бенефициенти и представители на бизнеса от бранша на тема “Проучване и анализ на състоянието на връзките между науката и бизнеса, изясняване на нуждите на бизнеса от иновационни продукти. Споделяне на опит в работата с ИТ сектора. Сближаване на ВУ и БАН с бизнеса”;
- За първата година от изпълнението на проекта са реализирани 83 публикации на крайните бенефициенти, подпомогнати финансово от Оперативна програма “Развитие на човешките ресурси”. Значителна част от тях са на международни форуми;
- Всеки краен бенефициент получи и непосредсвена финансова подкрепа под формата на стипендия за подпомагане на своята научно изследователска дейност.

Управителният съвет на проекта е убеден, че вложенията, които Оперативна програма “Развитие на човешките ресурси” прави за подкрепата на докторанти и млади учени са много навременни и несъмнено от полза за развитието на българската наука.

За контакти:
доц. д-р инж. Георги Стоянов Тодоров
ВТУ „Св. Св. Кирил и Методий” – В. Търново
Координатор на проекта

ИЗИСКВАНИЯ ЗА ОФОРМЯНЕ НА СТАТИИТЕ НА АВТОРИТЕ ЗА СПИСАНИЕ "КОМПЮТЪРНИ НАУКИ И ТЕХНОЛОГИИ”

I. Ръкописите на статията се представят разпечатани в два екземпляра (оригинал и копие) в размер до 6 страници формат А4 и като файл на адрес: Технически университет – Варна, катедра “Компютърни науки и технологии”, ул. Студентска 1, 9010 Варна.

II. Текстът на статията трябва да включва: УВОД (поставяне на задачата) ИЗЛОЖЕНИЕ (изпълнение на задачата), ЗАКЛЮЧЕНИЕ (получени резултати) , БЛАГОДАРНОСТИ към сътрудниците, които не са съавтори на ръкописа (ако има такива), ЛИТЕРАТУРА и адрес за контакти, включващ: научно звание и степен, име, инициали, фамилия, организация, поделение(катедра), e-mail адрес.

III. Всички математически формули трябва да са написани ясно и четливо (препоръчва се използуване на Microsoft Equation). Номерацията на формулите се дава вдясно от началото на реда. Текста на формулите се позиционира в средата на реда. Експоненциалните функции се изписват чрез знака exp.

1. Текстът трябва да бъде въведен във файл във формат WinWord 2000/2003 със шрифт Times New Roman. Форматирането трябва да бъде както следва:
2. Размер на листа - А4, полета - ляво - 20мм, дясно - 20мм, горно - 15мм, долно - 35мм, Header 12.5мм, Footer 12.5mm (1.25см).
3. Заглавие - (на български език, размер на шрифта 16, bold).
4. Един празен ред, размер на шрифта 14, нормален.
5. Имена на авторите - име, инициали на презиме, фамилия, без звания и научни степени, размер на шрифта 14, нормален.
6. Два празни реда, размер на шрифта 14.
7. Резюме -(на български език, размер на шрифта 11) - до 8 реда, нормален.
8. Заглавие (на английски език, размер на шрифта 12) – bold.
9. Един празен ред, размер 11.
10. Имена на авторите (на английски език, размер на шрифта 11) – нормален.
11. Един празен ред, размер 11.
12. Резюме - на англиийски език, до 8 реда размер на шрифта 11, нормален.
13. Основните раздели на статията (Увод, Изложение, Заключение, Благодарности) се форматират в едноколонен текст както следва:

n. Наименование на раздел или на подраздел, размер на шрифта 12, удебелен, центриран.
o. празен ред, размер на шрифта 12;
p. Текст, размер на шрифта 12, нормален, отстъп на първи ред на параграф – 10 мм; разстояние от параграф до съседните (Before и After) за целия текст – 0.
q. цитиране на литературен източник - номер на източника от списъка в квадратни скоби;
r. Номерация на формулите: дясно подравнена, в кръгли скоби.
s. Графики:. центрирани, разположение спрямо текста: “Layout: In line with text”.
t. Литература – всеки литературен източник се представя с: номер в квадратни скоби и точка, списък на авторите (първият автор започва с фамилия, останалите – с име), заглавие, издателство, град, година на издаване.
u. За контакти: научно звание и степен, име, презиме (инициали), фамилия, организация, поделение (катедра), e-mail адрес, с шрифт 11, дясно подравнено.

Образец за форматиране (компресиран Word файл) можете да изтеглите от адрес http://kst-forum.netfirms.com/Spisanie_Obrazec.zip

Този брой от списанието се отпечатва със средства от научно-изследователски проект, финансиран целево от държавния бюджет.