Time Warp-algoritmen

F R E D R I K J Ä F V E R T

KTH Datavetenskap
och kommunikation

Examensarbete
Stockholm, Sverige 2010

Time Warp-algoritmen

F R E D R I K J Ä F V E R T

Examensarbete i datalogi om 15 högskolepoäng
vid Programmet för datateknik
Kungliga Tekniska Högskolan år 2010
Handledare på CSC var Henrik Eriksson
Examinator var Mads Dam

URL: www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2010/jafvert_fredrik_K10071.pdf

Kungliga tekniska högskolan
Skolan för datavetenskap och kommunikation

KTH CSC
100 44 Stockholm

URL: www.kth.se/csc

Abstract

A minor investigation of the Time Warp algorithm, its pitfalls, developments and applications. Some modifications are described as well.

Sammanfattning

En mindre littteraturstudie som beskriver TimeWarp-algoritmen, dess fallgropar samt inom vilka moderna tillämpningar algoritmen används. Några varianter och framtida utveckling beskrivs också.

Innehåll

Kapitel 1

Bakgrund

1.1 Simulering

Att simulera någonting är att modellera en process över en viss tidsperiod. Skälen för att simulera en process kan vara många, men det är oftast för att få insikt i ett förlopp på annat sätt skulle varit svårt eller kostsamt att få reda på. Fördelar med att simulera är det även ger möjlighet att variera tidsflödet och eller titta närmare på andra variabler vilka kan underlätta identifierandet av begränsningar eller beteenden i systemet. Det som dock är värt att notera är att modell-skapandet kan vara en komplicerad process för att få med de för processen nödvändiga delarna samtidigt som resultatet från simuleringen kan vara svårtolkat.

För att simulera händelseförlopp finns det olika angreppssätt för att modellera tidens gång, dels varianter som stegar en konstant längd och de som stegar en variabel längd beroende på en yttre faktor.

1.2 Diskret händelsesimulering

Diskret händelsesimulering är en simuleringsmetod med variabel tidsteglängd, där steglängden bestäms utifrån tidpunkten för nästa kända händelse i simuleringen. På detta sätt kan simuleringar av modeller vars händelser har en händelsefördelning som inte är jämnt fördelad över tidsperioden som skall simuleras spara tid genom att inte simulera händelselös tid. Det är alltså värt att notera att om händelsefördelningen är jämn över tidsintervallet som skall simuleras samt händelserna ligger tätt relativt den tidsskala som används så finns det ingen vinst i tid med den diskreta simuleringen jämfört med den kontinuerliga [2].

Ett fall där diskret händelsesimulering skulle lämpa sig är om man är intresserad av att simulera vad en person äter under ett dygn så kan man förståss låta simuleringen ticka upp en tidsenhet i taget under hela dygnet, även under tider på dygnet då det inte händer någonting som när personen till exempel sover varpå man får en stor del av simulerad tid där ingenting händer i simulationen. Detta ger arbete i onödan då varje tidssteg genererar administrativt arbete för simulationen.

I stället kan man låta simulationen stega framåt med olika stora tidssteg beroende på när personen äter. Vi börjar på frukost och när man är färdig med att äta frukost så hoppar simulationen fram i tiden till lunch istället för att simulera minut för minut under tiden då personen inte äter någonting. Detta sparar oss då tid och trafik i form av administrativa meddelanden.

1.3 Parallell diskret händelsesimulering

Modellen som simuleras kan ibland delas upp i flera små delmodeller som kan köras på olika processorer. Då bör man välja att dela upp modellen för simuleringen i mindre delar av systemet.

Om vi vill simulera vad anställda gör på ett kontor kan vi dela upp denna simulering så att varje delsimulering motsvarar en anställd. Den anställda gör sina göromål under dagen i godan ro och så fort som möjligt efter egen förmåga. När den anställda behöver prata eller gör någonting som direkt kommer att påverka de andra anställda med någon så skickar processen ett meddelande till de delsimuleringarna som påverkas.

1.4 Optimistisk parallell diskret händelsesimulering

Om vi även tillåter delsimuleringarna att ha individuell tidräkning och olika steglängder kan alla delsimuleringar vara sysselsatta större delen av tiden då de kan behandla händelser som är relevanta för dem istället för att vänta på att tiden skall komma för en för dem relevant händelse.

Med individuell tidräkning finns två alternativ för beteendet. Antingen uppehålls så kallad lokal kausalitet, det vill säga, delsimuleringar får inte gå längre fram i tiden än till dess att man är säker på förutsättningarna för delsimuleringen. Detta går under namnet *konservativ sykroniserings strategi*. Den andra strategin bygger istället på att varje delsimulering får beräkna en förväntad framtid utifrån de förutsättningar som är kända vid tidpunkten, och ifall förutsättningarna förändras genom resultat från andra delsimuleringar återställs delsimuleringen till den tidpunkt där förutsättningarna ändrades och fortsätter på nytt men beräknar nu den nya förväntade framtiden med utgångspunkt i de förändrade förutsättningarna. Då talar man om *optimistisk syknronisering strategi*.

Kapitel 2

Time Warp

Time Warp tillhör de optimistiska parallella händelsesimuleringsalgoritmerna. Den introducerades först 1982 av Jefferson och Sowizral [6] och var tänkt att snabba upp den dåvarande tidens simulering av diskreta modeller, i synnerhet realtids stridssimulering.

Detta åstakoms genom att dela upp och parallellisera simuleringsarbetet över flera processorer samt låta alla dessa delsimulationer beräkna den förväntade framtiden var och en för sig och hoppa tillbaka i tiden om den förväntade framtiden inte inträffade. Detta påminner om begreppet *branch prediction* inom processorarkitektur [4].

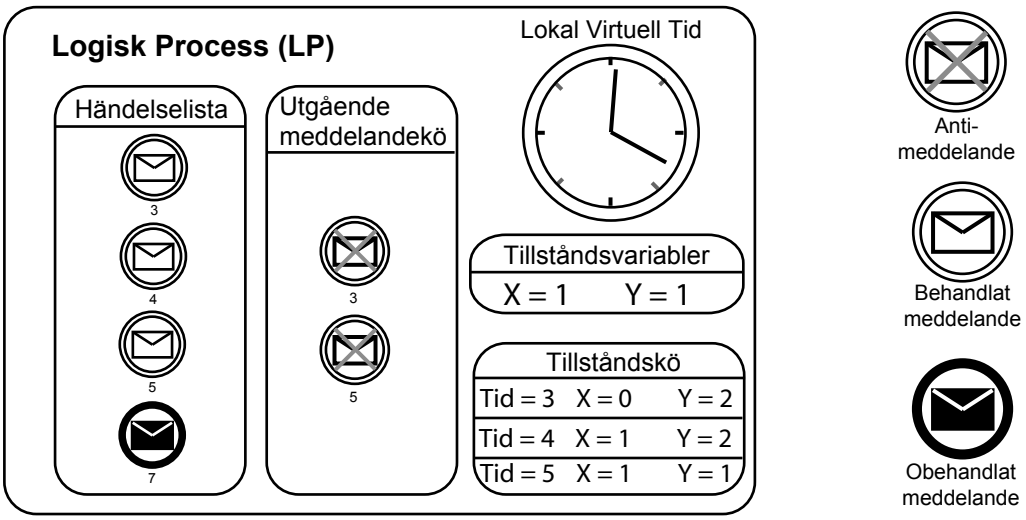
Time Warp konceptet går att dela upp i lokala och globala kontrollmekanismer, där de lokala kontrollmekanismen kontrollerar de logiska processerna och den globala kontrollmekanismen håller reda på simuleringen i stort [5].

2.1 Logiska Processer

De logiska processer som modellen delas upp i kan beskrivas som små sekventiella delsimuleringar och består av händelselista, tillståndsvariabler, tillståndskö, virtuell tid och utgående meddelandekö. Tillsammans utgör dessa komponenter den lokala kontrollmekanismen.

2.1.1 Händelselista

För att hålla ordning på framtida och historiska händelser så har varje logisk process en händelselista, dels för att veta vilka händelser som har behandlats och vilka som skall behandlas i framtiden. Denna händelselista används även för inkommande meddelanden och börjar fylld med de händelser som är kända vid simuleringens start men utökas allteftersom simuleringen går och andra logiska processer påverkar händelseförloppet.



Figur 2.1. Uppbyggnaden av en logisk process

2.1.2 Tillståndsvariabler

De egenskaper som den logiska processen har vid den nuvarande lokala virtuella tiden och vilka värden är tilldelade.

2.1.3 Tillståndskö

Tidigare tillstånd som den logiska processen har haft, sparade med tidsstämpel så det lätt ska gå att återgå till ett tidigare tillstånd vid en eventuell rollback.

2.1.4 Lokal Virtuell Tid

Varje logisk process har en egen tidräkning, som håller reda på var i tiden som den logiska processen befinner sig, och på så sätt relaterar till andra logiska processer i kommunikationen med dem. Denna lokala virtuella tid stegar sig framåt med variabel steglängd som beror av tidpunkten för nästa händelse i händelselistan.

2.1.5 Utgående meddelandekö

För att kommunicera förändringar i simuleringen mellan de logiska processerna använder man sig av tidsstämplade meddelanden och för att ha möjlighet att återställa effekterna av ivägskickade meddelanden så sparas ett 'anti-meddelande' för varje meddelande som har skickats iväg i den utgående meddelandekön. Om den logiska processen sedan behöver återställas så kommer den att skicka iväg alla anti-meddelanden som finns i kön med högre tidsstämpel än den tid som den logiska

processen återställdes till och på det viset återställa de förändringar som den felaktigt förväntade framtiden innebar.

2.2 Rollback

Vid det tillfälle då en logisk process gör beräkningar som påverkar sin omgivning så skickas meddelande till de berörda processerna. Om de berörda processerna skulle befinna sig på en lokal tid som är längre fram i tiden än ursprungsprocessen och den har gjort beräkningar som baseras på ett ursprungsvärde som förändras av det inkommande meddelandet så kommer mottagarprocessen att behöva stega tillbaka sin tid till samma tid som avsändarprocessen som har för att sedan beräkna en ny framtid som baserar sig på den nya informationen.

Det meddelande som sätter igång en rollback går under benämningen eftersläntarmeddelande (*straggler-message*), och är som vilket meddelande som helst förutom att det som skiljer den från andra meddelanden är att dess tidsstämpel ligger långt bak i tiden jämfört med var den mottagande logiska processen ligger just nu, vilket ofta resulterar i att ett eftersläntarmeddelande utlöser en våg av antimeddelanden från den utgående meddelandekön på den logiska process som tagit emot det.

2.3 Global Virtuell Tid

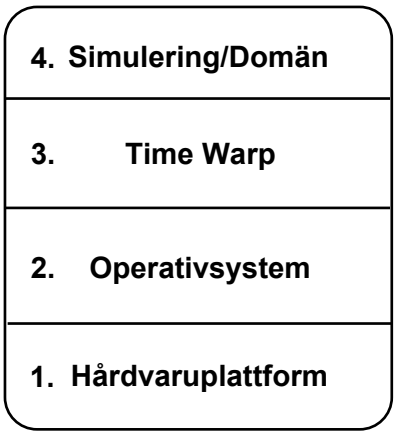
Den globala virtuella tiden (GVT) är en tidsangivelse som avser tiden för den delsimulering som hunnit kortast, det vill säga den lägsta lokala virtuella tiden i hela simuleringen.

Problemet med att spara alla tidigare tillstånd är att det tar enormt mycket plats, för att lösa det så kom man fram med GVT, den globala tidsvariabeln, detta för att veta vad för delar av de undansparade tillstånden som inte längre behövs, detta eftersom alla delsimulationer redan har passerat dessa tillfällen och simulationen aldrig kommer att hoppa tillbaka längre än detta. Denna process kallas *fossil collection* där ett undansparat tillstånd referas till som ett fossil och har fått många olika förslag på lösningar i nyare implementationer av TimeWarp.

Utöver detta användningsområde så används även GVT för att säkert kunna utföra utföra oåterkalleliga I/O operationer utan att riskera att korrumpera simuleringen.

2.4 Problem och Fallgropar

En parallel diskret händelsesimulering är en komplex mjukvara och det finns mycket som kan gå fel vid väldigt många olika tillfällen och på många olika nivåer. De olika nivåerna sträcker sig från simuleringslagret där orimligheter enligt modellen som skall simuleras kan uppstå ned igenom Time Warp-lagret och vidare ned till operativsystemslagret och hårdvarulagret. Beroendes på vart ett fel uppstår så kan



Figur 2.2. De olika nivåerna i en simuleringsplattform

det ge olika svåra konsekvenser för simuleringen och dess slutresultat och olika möjligheter att upptäcka att någonting verkligen har gått fel.

2.4.1 Orsaker till fel

Som vid all datorbehandling kan vissa fel uppstå på grund av dålig indata. I det här fallet betyder det att den indata som simulationen givits förutsätter enligt modellen omöjliga tillstånd, vilket i bästa fall ger fel i simuleringsnivån på simulationsplattformen men i sämre fall kan ge fel på operativsystemsnivå om man till exempel försöker ta fler äpplen än vad som finns i en hög och på så sätt orsakar minneskorruption när simuleringsplattformen kommunicerar vidare ner vad som skall utföras.

Fel kan även uppstå vid implementationen av Time Warp mekanismen och dess protokoll, vilket i sin tur kan ge väldigt svårspårade fel eftersom simuleringarna inte uppför sig som de ska men inte nödvändigtvis behöver resultera i orimliga resultat.

Men om man väljer att bortse ifrån dessa relativt oexotiska fel så finns det en hel uppsjö av fel som uppstå under körning och som kan vara väldigt komplexa att reda ut, åtgärda och skydda sig mot.

2.4.2 Symptom

Jag har valt att kategorisera fel beroende på var deras symptom uppträder, det vill säga var man upptäcker dem. De kan ha sitt ursprung i en annan kategori.

	Samma LP	Annan LP
Nutid	här och nu	där och nu
Framtid	här och sen	där och sen

Figur 2.3. Var och när problem upptäckts

Plattformsfel

Plattformsfel är fel som syns och ger systemfel eller andra fel som går att upptäcka i operativsystemsnivån i form av felsignaler. Dessa fel kan dels bero på index-fel och segementeringsfel men även borttagning av objekt från tomma datastrukturer. Det som är gemensamt för dem är att de genererar något slags operativsystemsfel och på så sätt ger sig till känna.

Time Warp fel

I denna kategori så hamnar de fel som mer eller mindre beror på Time Warp-plattformen i sig, det kan röra sig om oändliga loopar eller rekursiva funktioner utan slut.

Domänfel

Utöver plattformsfel och Time Warp fel så finns även kategorin domänfel, vilka innefattar de fel som ger omöjligheter/orimligheter inom området som simuleras. Felen i sig kan ha ursprung och vara orsakade av plattformsfel men även bero på felaktiga indata eller en felaktig modellering.

2.4.3 Lokalitet

För att klassificera vart ett problem upptäckts och vart det uppstått så går det att placera fel i både tid och rum, för även fast ett fel har upptäckts så behöver det inte nödvändigtvis betyda att felet uppstod där det upptäcktes utan det upptäckta felet kan även vara ett följdfel. Ett exempel på följdfel skulle kunna vara någon som tar ett äpple för mycket ur en korg och sedan uppstår fel när någon som egentligen skulle ha fått ett äpple istället får fel efter att ha försökt att ta upp en frukt som inte fanns. Vilket då skulle hamna i den nedre högra kvadranten i diagrammet för fel-lokalitet.

2.4.4 Åtgärder

Olika fel kräver olika åtgärder, och även om det inte går att bygga bort allting så finns det många metoder för att komma tillrätta med en stor del av felen som kan uppstå.

När det gäller indata så går det till viss del att förbehandla indata för att se att de inte bryter mot de förutsättningar som gäller inom den modell som simuleringen gäller. Och för att se till att Time Warp mekanismen implementeras på ett väl fungerande sätt så finns det till exemepel ett ramverk för att verifiera att kausalitet följs av den uppbyggda plattformen [3].

När det kommer till runtime-felen så är det viktigt att notera att även fast ett fel har upptäckts så är det inte säkert att det är fel i simuleringen och att den simuleringen måste avbrytas [4]. Den åtgärd som kan vara lämplig är att vänta och se ifall felet åtgärdas genom en rollback. Detta resulterar i att vi väntar på att GVT ska passera den nuvarande lokala tiden för den logiska processen innan vi bryter hela simuleringen för att fel har uppstått.

2.5 Tillämpningsområden

Användningsområden där simuleringar med timewarpalgoritmen används eller har använts innefattar bland annat realtids stridssimulering, datorarkitekturer, flygledning, datorkommunikationsnätverk, Supply Chain Management, P2P Spel [9].

2.6 Utveckling och tillägg

Det grundläggande konceptet för Time Warp algoritmen har utvecklats med åren och modifikationer på den har reviderats och uppdaterats.

Det som introducerades i systemet GTW (Georgia Time Warp) och har gjort stora förbättringar då den använts inom stelkroppssimulering [7] och har även på senare tid använts i datorspels fysikmotorer vid simulering av mjuka kroppar och med flera tidsskalor.

Området fossil collection har även utvecklats vidare åt många olika riktningar såsom *batch fossil collection* och *on-the-fly fossil collection* [4]

Time Window

Även fast vi väljer att begränsa vilken information vi sparar bakåt i tiden kan vi få problematik med minnesbehov, beroende på hur snabbt olika logiska processer går. En lösning är att sätta en gräns på hur långt in i framtiden som logiska processer får gå relativt GVT, detta för att se till att alla logiska processer ligger inom ett tidsintervall och på så sätt begränsar all information som behöver lagras i fall det sker en rollback. Detta leder i sin tur till frågeställningen angående vilken storlek på intervallet för det så kallade tidsfönstret skall ha för att optimalt ta vara på de resurser som finns tillhandahållna.

Reverse computation

Istället för att spara tidigare tillstånd för att ha möjlighet att återställa till en tidigare tidpunkt så har även en metod för att återberäkna till ett tidigare tillstånd

uppfunnits och man på detta sätt kan spara utrymme [1] [8].

2.7 Sammanfattning

2.7.1 Sammanfattning

Det är ett stort område och det är fortfarande under utveckling och frammarch. I och med att utvecklingen går frammåt går uppskalningen av Time Warp implementationerna mot fler och fler processorer vilket leder till nya uppfinningar inom området för att se till att hålla nere de administrativa meddelandena vilket leder till kombinationer av de optimistiska och konservativa kommunikationsstrategier. Detta i sin tur fordrar mer analys under modelleringssteget då man behöver insikt i vilken dynamik som olika delar av modellen har inneboende, någonting som kan behöva åtgärdas med hjälp av iterativ modellutveckling för att få kännedom om vilka delar som lämpar sig för optimistisk respektive konservativ kommunikation.

Litteraturförteckning

[1] Christopher D. Carothers, Kaylan S. Perumalla, and Richard M. Fujimoto. Efficient optimistic parallel simulations using reverse computation. In *Proceedings of the thirteenth workshop on Parallel and distributed simulation*, PADS '99, pages 126–135, Washington, DC, USA, 1999. IEEE Computer Society.

[2] Henrik Eriksson. Event-controlled or time-controlled simulation. *Nordic Computer Symposium, Stockholm*, 1964.

[3] P. Frey, R. Radhakrishnan, H. W. Carter, P. A. Wilsey, and P. Alexander. A formal specification and verification framework for time warp-based parallel simulation. *IEEE Trans. Softw. Eng.*, 28:58–78, January 2002.

[4] Richard M. Fujimoto. *Parallel and Distribution Simulation Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999.

[5] D. Jefferson, B. Beckman, F. Wieland, Blu L., and M. Diloreto. Time warp operating system. *SIGOPS Oper. Syst. Rev.*, 21:77–93, November 1987.

[6] David. Jefferson. *Fast concurrent simulation using the time warp mechanism, part 1 local control / David Jefferson [and] Henry Sowizral*. Washington : United States Air Force, 1982.

[7] Brian Mirtich. Timewarp rigid body simulation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 193–200, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[8] Andriy Naborskyy and Richard M. Fujimoto. Using reversible computation techniques in a parallel optimistic simulation of a multi-processor computing system. In *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation*, PADS '07, pages 179–188, Washington, DC, USA, 2007. IEEE Computer Society.

[9] Stefan Tolic and Helmut Hlavacs. *A Testbed for P2P Gaming Using Time Warp*, pages 33–47. Springer-Verlag, Berlin, Heidelberg, 2009.