Реализация на език за разпределена симулация

Христо Вълчанов

Implementation of a distributed simulation language: *Simulation is becoming more widely used in studying the behavior of complex systems in many areas of modern life. Particularly relevant is the need to have effective tools of simulation (linguistic and instrumental) to facilitate and accelerate the development of simulation models. In this paper the implementation aspects of an object-oriented language for distributed simulation, called SIMOPAL, are presented and discussed.*

Key words: *TimeWarp, Distributed Simulation, Simulation Language.*

ВЪВЕДЕНИЕ

Симулацията намира все по-широко приложение при изследване поведението на сложни системи в множество области на съвременния живот. Създаването на добри симулационни модели е съпътствано от редица трудности. Цената на проектирането и разработването на сложни модели е понякога съизмерима с цената на изграждане на реална система. Същевременно, не всички изследователи са експерти в областта на симулацията, което води до наличието на проблеми в процеса на създаваните от тях симулационни модели. Изследванията показват, че особено актуална е необходимостта от наличие на ефективни средства за симулация (лингвистични и инструментални), които да улеснят и ускорят разработването на симулационни модели [5,7]. Езиците за симулация се явяват именно такъв мощен инструмент, предоставящ богати функционални възможности на базата на добре дефинирани езикови конструкции. Все по-нарастващите изисквания към ефективността на процеса на симулация налагат необходимостта от разработване на високопроизводителни, гъвкави и икономически ефективни изпълнителни среди на езиците за симулация.

В [6] е представен език за разпределена симулация SIMOPAL за моделиране на дискретно-събитийни системи. За разлика от редица типични представители на симулационните езици, като Parsec, Apostle, ModSim, SimJava [7], които са базирани на концепцията за логически процеси [2], SIMOPAL е проектиран на основа на модела на Actors [1], което дава възможност за постигане на по-високо ниво на абстракция при описание на моделираната система.

В настоящия доклад са представени някои аспекти на реализацията на езика в разпределена изчислителна среда- мрежа от работни станции.

ЕЗИК ЗА СИМУЛАЦИЯ

Представеният в [6] език за разпределена симулация SIMOPAL е базиран на Actors [1], като обобщен модел на обектно-базирана паралелна система. Обектът е абстрактно описание на самостоятелен обработващ елемент, притежаващ локална памет. Единственото допустимо взаимодействие между обектите е асинхронният обмен на съобщения. Базовият модел на Actors е разширен чрез въвеждане на класове. Обектите се разглеждат като екземпляри на класове. Класът, от своя страна, задава шаблона на структурата и поведението на своите екземпляри. Развитието на модела в процеса на симулация се базира на промяна на локалните състояния на екземплярите на класовете под въздействието на настъпващи в модела събития. Промяната на състоянието на обект представлява извършване на преходи между множеството от неговите локални състояния. Локален преход се описва с помощта на правило, дефинирано като правило на поведение. Правилата на поведение се представят във вида:

<лява_страна> <дясна_страна>

,където и двете части представляват изрази (терми).

Лявата страна е терм, описващ настъпването на събитие, а дясната страна е

терм, описващ следващото състояние вследствие на това събитие. Тя въздейства върху конфигурацията на модела.

Множеството от всички правила за даден клас се дефинира като база с правила на поведение на екземплярите на този клас. Настъпване на локален преход в състоянието на обект се интерпретира формално като удовлетворяване само на едно правило от тази база. Удовлетворяването на правило се свежда до изпълнението на действията, съдържащи се в неговата дясна страна: планиране на ново събитие в симулационния модел, създаване на нов обект и/или промяна на атрибутите на обекта.

Всеки екземпляр на клас притежава локална памет, в която се съхраняват атрибутите, формиращи неговото локално състояние. Множеството от локалните състояния на всички екземпляри определя моментното състояние на цялата моделирана система.

На фиг.1 е показан програмен фрагмент, илюстриращ моделиран компонент – маршрутизатор, който изпълнява определени действия при получаване на събития.

```
typeclass Router () {                    // клас
    int data; bool processing;           // атрибути
    event Update { int data };           // събития
    event Data { int content };

    isevent (Update u) {                 // правило на поведение
        if (not full) {
                data = u.data;           // промяна на състоянието
                processing = true;
                // планиране на събитие Update за следващ маршрутизатор
                // след 10 единици модело време
                deposit(next_router, Update, 10);
        }
    };
    isevent (Data d) {                   // правило на поведение
        ...
    };
};
```

Фиг. 1 – Примерен програмен фрагмент на SIMOPAL

Примерът илюстрира декларативния характер на езика при описание на компонентите на системата. При симулация на системи, съдържащи хиляди компоненти, явното деклариране на единичен компонент е неефективно. Въвеждането на класове позволява дефиниране единствено на шаблоните на поведение на обектите на моделираната система. От друга страна, използването на правила на поведение дава възможност за повишаване нивото на абстракция при описание на развитието на реалната система.

СИМУЛАЦИЯ, БАЗИРАНА НА ДИСКРЕТНИ СЪБИТИЯ

Естествен подход при паралелните езици е реализирането на екземплярите на класовете като процеси. В случаите на симулация на системи, съдържащи хиляди моделирани обекти, това е неприемливо от гледна точка на ефективността на симулацията. Производителността на симулацията силно ще се влоши вследствие на големите системни разходи за създаване и превключване на контекста на процесите. Намаляване на тези разходи може да се постигне чрез използване на нишки. Характерно обаче за тях е, че те имат недетерминистична натура [4]. Именно тази недетерминираност е нежелателна при провеждането на симулации, изискващи предсказване и повторяемост на експериментите.

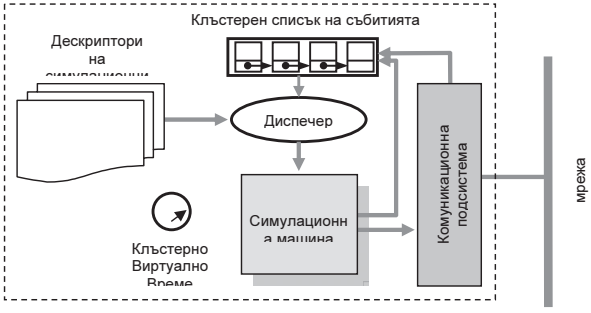
Алтернативен подход е симулация, базирана на дискретни събития [8]. Промяната в състоянието на моделираната система се отразява чрез настъпване на събития, реализирани като предаване на съобщения между обектите. Всяко съобщение съдържа информация, характеризираща типа на събитието и моделното време на неговото настъпване (времеви маркер).

За реализиране на процеса на симулация, събитията са подредени по времевите си маркери в специална структура, наречена „глобален списък на събитията". Тяхната обработката се извършва последователно от началото на списъка. В резултат на обработката на събитие, обектът, за когото е предназначено то, изпълнява определени действия, които могат да рефлектират в промяна на неговото състояние или изпращане на нови съобщения към други екземпляри, симулиращи настъпване на нови събития. Този начин на реализация е известен като „последователна симулация".

РАЗПРЕДЕЛЕНА СИМУЛАЦИЯ

Събитията, между които отсъства причинно-следствена връзка, могат да бъдат паралелно изпълнени. Това дава възможност за ускоряване на процеса на симулация чрез неговото разпаралелване между множество изчислителни възли. Проблемите, свързани със закъсненията в комуникационните среди, а оттам и настъпването на причино-следствени грешки в процеса на симулация, се решават чрез използването на специализирани протоколи за синхронизация. Един от най-разпространените е оптимистичният протокол TimeWarp [3]. Протоколът позволява симулационните обекти (СО) да обработват събитията по реда на тяхното получаване (пристигане на съответните съобщения). След обработване на събитие, локалното моделно време на обекта става равно на времевия маркер на събитието. Подходът допуска обработване на събития в последователност, различна от естественото им настъпване, което може да причини причинно-следствена грешка. Такава грешка настъпва при получаване на съобщение за събитие (*съобщение-нарушител*) с времеви маркер, по-малък от текущото локално моделно време на обекта. Процедурата по възстановяване изисква СО да върне своето локално състояние до момента преди получаването на съобщението-нарушител и да отмени всички изпратени до момента съобщения за планирани събития. За целта СО периодично съхранява своето състояние. Отмяната се реализира чрез изпращане на съответните и анти-съобщения. Процесът на възстановяване е известен под наименованието „връщане назад".

Изпълнителната среда на езика е реализирана като клъстерна структура (фиг.2). Всеки компонент от клъстера представлява самостоятелен симулатор, който се изпълнява като отделен процес в операционната система. В рамките на симулатора симулацията е последователна на базата на клъстерен списък на събитията. Между отделните симулатори е реализиран алгоритъма TimeWarp.



Фиг. 2 – Клъстерна структура на изпълнителната среда

КОМПИЛАТОР НА ЕЗИКА

При реализацията на езиците за разпределена симулация основен фактор е бързодействието на симулационната система. Редица съвременни езици за

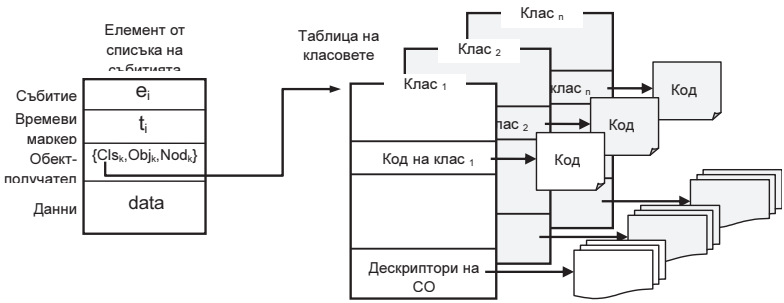
симулация [7] са разработени на базата на платформи като Java или .NET, което определя и използването на интерпретатори за изпълнение на симулацията. Това, в комбинация с разпределения характер на изпълнение и наличието на междупроцесорна комуникация, е фактор, който ограничава производителността на симулацията.

В представената симулационна система е приложен подходът за изпълнение на машинен код вместо интерпретатор. Директното транслиране на езиковите конструкции в машинен код не е тривиална задача и изисква разработване на компилатор за различни операционни системи и архитектури. С цел постигане на гъвкавост и преносимост, компилаторът на езика генерира изходен ANSI C програмен код под формата на текстов файл. Този C код може да бъде компилиран в машинен код за определена архитектура посредством стандартен C компилатор на съответната операционна система. На този етап програмата на езика SIMOPAL се свързва с библиотечните модули на разпределената изпълнителна среда. Преносимостта може да бъде постигната чрез използването на различни библиотечни модули за определени архитектури.

ПРЕДСТАВЯНЕ НА СИМУЛАЦИОННИТЕ ОБЕКТИ

Всеки СО реализира правилата на поведение на екземпляр от определен клас на SIMOPAL. Поддържането в паметта на програмния код на всеки екземпляр е изключително неефективно при наличието на хиляди обекти в рамките на клъстер. Решаването на този проблем е постигнато посредством прилагането на специална организация на представянето на СО в клъстера.

Отделните клъстери поддържат в своята памет програмния код на правилата на поведение на дефинираните в програмата на SIMOPAL класове. Правилата за поведение са транслирани от компилатора на езика във вид на функции, имащи като аргументи указатели към памет, съответстваща на атрибутите на екземпляр от съответен клас. Според модела на разпределена симулация [6], обект се представя посредством наредена тройка {Clsk,Objk,Nodk}, където Clsk е номер на клас, Objk е номер на екземпляр на този клас и Nodk е номер на клъстер. В клъстерите се поддържат таблици на класовете, всеки елемент от които съдържа указател към функцията, реализираща поведението на екземплярите на класа и таблица с дескрипторите на всички създадени екземпляри от този клас в рамките на текущия клъстер (фиг.3).



Фиг. 3 – Вътрешна организация на клъстер

Тази организация редуцира значително необходимия обем памет за съхраняване на кода на всеки отделен обект. Изпълнението на обектите се реализира посредством *симулационна машина*, която се управлява от диспечер. За ускоряване на процеса на търсене на дескрипторите на обектите, дескрипторните таблици на екземплярите за всеки клас са хеширани, като ключът за достъп до тях

се формира на базата на номер на обекта в съответния клас.

Развитието на даден обект се симулира чрез изпълнение на кода на правилата на поведение на съответния клас от симулационна машина. Всяко правило се активира, ако обектът е получил съобщение за настъпването на определено събитие. Основните действия на обекта при интерпретирането на правилата на поведение се свеждат до обработка на лявата част на всяко правило и евентуалното изпълнение на действията, описани в неговата дясна част.

Обработката на лявата част на правило осъществява съпоставяне на текущото получено съобщение с кода на обекта. Като резултат на този етап се определя дали да се разреши прилагането на дясната част на правило (код). Нека текущото съобщение е $M(P_k)$, където M е идентификатор на съобщението и P_k са неговите параметри. Текущото правило има лява част във вида (M_R P_R). За да е удовлетворено правило, то трябва да се отнася за текущото съобщение, т.е. M_R=M. Параметрите на съобщението се копират от буфера във временната променлива P_R, която се явява контейнер, с чиято помощ се осъществява достъп до параметрите на съобщението. P_R има локално значение в рамките на текущото правило на поведение.

ЗАКЛЮЧЕНИЕ

Голям практически интерес представлява създаването на езици за симулация и поддържащи средства за разработка, позволяващи описанието на модела да бъде независимо от използвания метод и базовата архитектура. Изпълнителната среда играе важна роля за ефективността на процеса на симулация. Разработването на високопроизводителни, гъвкави и икономически ефективни изпълнителни среди е особено актуално при симулацията на сложни системи. В настоящия доклад са представени някои аспекти на реализацията на симулационен език SIMOPAL в разпределена изчислителна среда. Като насоки за бъдеща работа могат да бъдат посочени адаптиране на изпълнителната среда върху grid-платформи, както и реализиране мигриране на симулационните обекти между изчислителните възли.

ЛИТЕРАТУРА

[1] Agha, G. Actors: A Model of Concurrent Computation in Distributed Systems. MIT Press, Cambridge, 1986.

[2] Banks, J., J.S. Carson, B.L. Nelson. Discrete-Event System Simulation (5th ed.). Upper Saddle River, Prentice Hall, 2009.

[3] Jefferson, D.R. Virtual Time. ACM-TOPLAS, 7(3), 1985, 404–425.

[4] Lee, E.A. The Problem with Threads. IEEE Computer, 39(5), 2006, 33-42.

[5] Parumalla, K. Principles of Advanced and Distributed Simulation. Simulation, 2009, 227-228.

[6] Valchanov, H. An Actor Based Language for Distributed Simulation. Proc. of Int. Conf. Automatics and Informatics'11, 2011, B-367 – B-370.

[7] Yeo, C. S. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. Soft. Pract. Exper., 2004, 653-673.

[8] Zeigler, B.P., H. Praehofer, T.G. Kim. Theory of Modeling and Simulation. Academic Press (2nd ed.), 2000.

За контакти:

гл. ас. д-р инж. Христо Георгиев Вълчанов, Катедра "Компютърни науки и технологии", Технически университет - Варна, тел.: 052/383424, e-mail: hristo@tu-varna.bg

Докладът е рецензиран.