Kỷ yếu Hội nghị Khoa học Quốc gia lần thứ IX "Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR'9)"; Cần Thơ, ngày 4-5/8/2016
DOI: 10.15625/vap.2016.000107

XỬ LÝ SONG SONG MÔ PHỎNG ĐA TÁC TỬ GIS BẰNG PHƯƠNG PHÁP CHIA MẢNH

Nguyễn Hồng Quang[1], Hồ Tường Vinh[1], Nguyễn Mạnh Hùng[2]

[1]Phòng nghiên cứu MSI, Viện Quốc tế Pháp ngữ – ĐHQGHN
[2]Khoa CNTT-TT, Học viện Bưu chính Viễn thông Hà Nội
nguyen.hong.quang@ifi.edu.vn, ho.tuong.vinh@ifi.edu.vn, nmhufng@yahoo.com

TÓM TẮT *— Những công cụ mô phỏng đa tác tử trên môi trường thông tin địa lý (Geographical Information System – GIS) như GAMA (GIS Agent-based Modeling Architecture) thường cho phép chạy một mô phỏng chỉ trên một máy chủ. Với những mô phỏng hệ thống lớn phức tạp với số lượng lớn tác tử tham gia (hàng trăm ngàn hay hàng triệu) thì các máy PC thông thường sẽ không tải nổi hoặc thời gian xử lý sẽ bị kéo dài đến mức không chấp nhận được. Những phương án song song hóa công cụ mô phỏng để tăng tốc độ xử lý trên các máy chủ song song hoặc đa nhân đem lại kết quả hạn chế và đòi hỏi phải xem lại kiến trúc và thiết kế của công cụ mô phỏng, chương trình cũng như ngôn ngữ mô phỏng đi kèm.*

Chúng tôi đề xuất một cách tiếp cận trực quan độc lập với công cụ và ngôn ngữ mô phỏng dựa trên đặc thù môi trường thông tin địa lý của bài toán mô phỏng: chia môi trường địa lý mô phỏng đã cho thành các mảnh nhỏ cùng với các tác tử đã được phân phối ban đầu trên nó và sử dụng nhiều máy chủ chạy song song cùng một chương trình mô phỏng nhưng mỗi máy chỉ phụ trách môi trường một mảnh. Tổng hợp các mô phỏng con này trong mối quan hệ tương hỗ lẫn nhau sẽ cho kết quả của bài toán mô phỏng toàn thể. Trong bài báo này, chúng tôi phân tích và liệt kê những mối liên hệ có thể giữa tác tử các vùng liền kề. Tiếp đó chúng tôi đề xuất phương án giải quyết những bài toán do những liên hệ này đặt ra. Một mô phỏng cụ thể áp dụng cách tiếp cận này, thực hiện trên công cụ mô phỏng đa tác tử GAMA sẽ chứng minh tính khả thi của tiếp cận và chỉ ra những vấn đề mà tiếp cận cần giải quyết để có thể ứng dụng trong thực tế.

Từ khóa *— Mô phỏng song song, đa tác tử, GIS, GAMA.*

I. MỞ ĐẦU

Trong những năm gần đây, mô hình hóa dựa trên tác tử (*agent-based modeling*) đã được sử dụng ngày càng rộng rãi để nghiên cứu các hệ thống phức tạp. Điều này xuất phát từ khả năng của tác tử cho phép thể hiện nhiều mức độ tương tác với những thể hiện môi trường chi tiết và phức tạp. Để đối phó với sự phức tạp gia tăng này, các công cụ mô hình hóa và mô phỏng mạnh mẽ đã được ra đời. Những năm qua đã chứng kiến sự phát triển của nhiều nền tảng dành riêng cho phát triển ứng dụng dựa trên mô hình tác tử. Trong số các công cụ này, nền tảng mô hình hóa và mô phỏng đa tác tử trên nền hệ thống thông tin địa lý GAMA (*GIS Agent-based Modeling Architecture*)[1] là một thí dụ tiêu biểu. Là một nền tảng được phát triển từ sự cộng tác giữa các nhà khoa học Pháp và Việt Nam tại phòng nghiên cứu Mô hình hóa và mô phỏng tin học các hệ thống phức tạp (MSI - *Modelisation et Simulation Informatique des systemes complexes*) thuộc Viện Tin học Pháp ngữ (nay là Viện Quốc tế Pháp ngữ, ĐHQGHN), GAMA nhằm mục đích hỗ trợ thiết kế các mô hình liên quan không gian địa lý, đa mô thức (*multi-paradigms*) và đa quy mô (*multi-scales*). GAMA đã được ứng dụng trong nhiều dự án nghiên cứu xung quanh các vấn đề nóng như dự báo, phòng chống thiên tai, thảm họa, dịch bệnh, v.v... [2].

Tuy nhiên, những công cụ mô phỏng dựa trên hệ thống thông tin địa lý như GAMA thường cho phép chạy một mô phỏng chỉ trên một máy chủ. Với những mô phỏng hệ thống lớn phức tạp với số lượng lớn tác tử tham gia (hàng trăm ngàn hay hàng triệu) thì các máy PC thông thường, kể cả có cấu hình cao, sẽ không tải nổi hoặc thời gian xử lý sẽ bị kéo dài đến mức không chấp nhận được.

Trong bài báo này, chúng tôi đề xuất một cách tiếp cận trực quan độc lập với công cụ và ngôn ngữ mô phỏng dựa trên đặc thù môi trường thông tin địa lý của bài toán mô phỏng: chia môi trường địa lý mô phỏng đã cho thành các mảnh nhỏ cùng với các tác tử đã được phân phối ban đầu trên nó và sử dụng nhiều máy chủ chạy song song cùng một chương trình mô phỏng, nhưng mỗi máy chỉ phụ trách môi trường một mảnh. Tổng hợp các mô phỏng con này trong mối quan hệ tương hỗ lẫn nhau sẽ cho kết quả của bài toán mô phỏng toàn thể.

Cấu trúc của bài báo như sau: mục tiếp theo phần mở đầu này sẽ đề cập các nghiên cứu liên quan đến chủ đề nghiên cứu. Sau đó sẽ là hướng tiếp cận của bài báo chia cắt môi trường mô phỏng và phân tích các vấn đề sẽ nảy sinh qua những mối liên hệ có thể giữa tác tử các vùng liền kề khi thực hiện mô phỏng. Tiếp đó, phương án giải quyết những bài toán này trên một bộ mô phỏng cụ thể là nền tảng GAMA sẽ được đề xuất, bao gồm những công cụ bổ sung mới và một qui trình áp dụng để " song song hóa " một mô phỏng đã có trên GAMA. Những giải pháp tối ưu cho việc thiết kế và cài đặt giải pháp cũng sẽ được đề cập nhằm xây dựng một prototype để chứng minh tính khả thi và hiệu quả của tiếp cận và chỉ ra những vấn đề mà tiếp cận cần giải quyết để có thể ứng dụng trong thực tế. Cuối cùng là một số kết luận của bài báo.

II. NGHIÊN CỨU LIÊN QUAN

Những nghiên cứu xung quanh việc song song hóa các ứng dụng mô phỏng dựa trên mô hình tác tử không có nhiều. Theo những tìm hiểu của chúng tôi về chủ đề này thì có vài cách tiếp cận sau đây:

F. Shimojo *et al.* [5] đã đề xuất một thuật toán " chia để trị " trên các lưới không gian thực để song song hóa ứng dụng mô phỏng động học phân tử dựa trên lý thuyết hàm trù mật để đạt được sự gia tăng tuyến tính số CPU xử lý khi số lượng phân tử gia tăng. Thuật toán đề xuất phân lớp theo không gian và được lập trình trên siêu máy tính song song. Ứng dụng rất đặc thù và không có liên quan đến mô phỏng đa tác tử. Cách tiếp cận của chúng tôi cũng theo ý tưởng chia nhỏ và có thể mở rộng tuyến tính khi lượng tác tử tăng.

Anne Hakansson [6] đề xuất một cách tiếp cận cho phép tự động tạo ra một phân cấp các tác tử trong một ứng dụng đa tác tử. Các tác tử tương tự được nhóm trong cùng một cấp nhằm tối ưu hóa tính địa phương và cải thiện đáng kể hiệu quả các tác vụ, giảm chi phí giao tiếp. Nhờ việc phân lớp, các nhóm tác tử có thể phân thành các tác vụ độc lập và có thể song song hóa. Cách tiếp cận có thể sử dụng cho khai phá dữ liệu, tìm kiếm tương tự. Không có khái niệm tác tử phụ thuộc vị trí và môi trường địa lý như trong loại mô phỏng đa tác tử mà chúng tôi hướng đến.

A. Nakano *et al.* [7] đề xuất một framework để mô phỏng nguyên tử diện rộng (hàng triệu đến hàng tỷ nguyên tử) dành cho phản ứng hóa học, cũng theo mô hình " chia để trị " chạy trên siêu máy tính cấp độ vài ngàn lõi. Ứng dụng cũng rất đặc thù và không có liên quan đến mô phỏng đa tác tử.

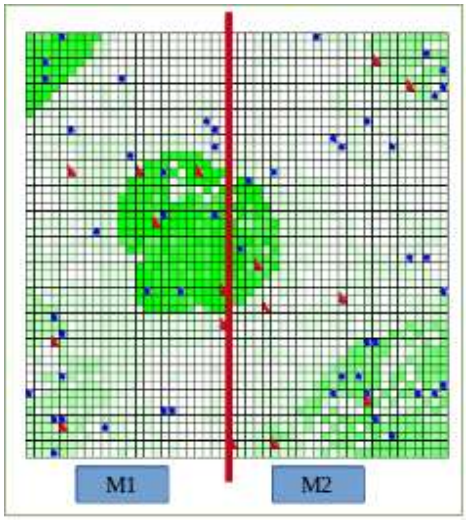
Chính trong GAMA, chủ đề song song hóa nền tảng mô phỏng để tăng tốc độ mô phỏng cũng đã được đề cập trong diễn đàn phát triển của dự án (https://github.com/gama-platform/gama/issues/738). Cách tiếp cận của các nhà phát triển đề xuất là tận dụng số lõi có trong bộ vi xử lý để khởi tạo một số *threads* song song nhằm tăng tốc độ xử lý. Nhóm đã đạt được một số kết quả bước đầu tích cực, tuy nhiên vẫn còn chưa đủ chín để công bố. Tiếp cận yêu cầu phải sửa trong lõi của GAMA và chỉ có thể cải thiện tốc độc tính toán nhưng không đáp ứng được nhu cầu xử lý bài toán mô phỏng với số lượng lớn tác tử vì ứng dụng vẫn chỉ chạy trên một máy PC.

Nhìn chung các tiếp cận đều hướng tới việc tăng tốc độ xử lý mô phỏng bằng cách tận dụng sức mạnh của các bó (*cluster*) hoặc siêu máy tính (*massivly parallel computer*), hoặc bằng cách tận dụng kiến trúc đa lõi trên các PC phổ dụng để tăng cường xử lý song song. Trên những nền tảng mô phỏng đa tác tử dựa trên môi trường địa lý như dạng GAMA, chúng tôi chưa thấy cách tiếp cận tương tự của mình.

III. PHƯƠNG PHÁP CHIA NHỎ MÔ PHỎNG ĐỀ XUẤT VÀ CÁC VẤN ĐỀ HỆ QUẢ CẦN GIẢI QUYẾT

Mô phỏng đa tác tử di động nói chung đều cần thiết một hệ tọa độ qui chiếu cho phép các tác tử xác định được vị trí (tọa độ) của mình trong thời điểm hiện tại cũng như điểm nó có thể di chuyển đến trong bước tiếp theo. Mô phỏng đa tác tử trên nền GIS cho phép các tác tử trong mọi thời điểm nhận biết được vị trí của mình trong môi trường (địa lý) theo một hệ tọa độ do GIS quy định. Tác tử di động có thể di chuyển trong không gian mô phỏng và có khả năng nhận biết biên giới, chướng ngại vật cũng như lối thoát khỏi không gian mô phỏng, căn cứ vào thông tin môi trường do GIS cung cấp và hành vi đã được lập trình.

Hình 1. Chia cắt vùng không gian mô phỏng thành 2 vùng



Việc chia cắt không gian mô phỏng thành 2 hay nhiều vùng liền kề có thể hiểu đơn giản là xác định một loại đường biên mới, ảo, phân tách giữa 2 vùng mô phỏng liền kề (xem Hình 1)

Trong Hình 1, không gian mô phỏng đã được chia làm 2 vùng liền kề. Đây là một ví dụ hết sức đơn giản để thể hiện ý tưởng chia vùng. Trong thực tế, một mô phỏng trên nền GIS thì biên giới giữa hai vùng liền kề nói chung sẽ không phải là một đường thẳng mà sẽ là tập hợp một hay nhiều đối tượng mà GIS cho phép nhận biết, được chỉ định làm biên giới (ví dụ 1 con sông, một chuỗi đường phố nối tiếp nhau, v.v...).

Ý tưởng được đề xuất ở đây là sau khi đã chia không gian mô phỏng thành các vùng liền kề thì mỗi vùng (cùng với các tác tử đã được triển khai trên nó) sẽ được phân phối cho các máy tính khác nhau trong cùng một bó máy (*cluster*) hoặc cùng một mạng cục bộ (LAN) tốc độ cao thực hiện song song. Các máy tính này phải được đồng bộ hóa theo bước mô phỏng. Kết quả mô phỏng của tất cả các máy trong bó sẽ được tập hợp về một điểm chung để hiển thị. Với ví dụ tại Hình 1, vùng bên trái sẽ được nạp cho máy M1 và vùng bên phải cho máy M2. Mỗi máy sẽ chỉ thực hiện hành vi và thay đổi thuộc tính cho các tác tử trong " lãnh thổ " của mình, tức là tải cho mỗi máy sẽ được giảm đi xấp xỉ một nửa so với tải ban đầu.

Yêu cầu tổng quát là kết quả mô phỏng phải không bị phụ thuộc vào việc mô phỏng được thực hiện trên 1 hay được chia thành nhiều máy song song. Nói cách khác, việc song song hóa mô phỏng phải trong suốt với người sử dụng.

Để thực hiện được ý tưởng đã nêu với yêu cầu tổng quát này, ngoài việc chia cắt " vật lý " môi trường mô phỏng ban đầu thành các các vùng liền kề như đã nêu trên, còn những vấn đề sau đây sẽ cần được giải quyết :

1. **Trao đổi thông tin giữa hai vùng liền kề**: tác tử nằm trong khu vực giáp ranh của 2 vùng liền kề phải nhận

được thông tin từ vùng bên cạnh như khi không có đường biên giới ảo phân cách vùng. Khu vực giáp ranh được định nghĩa là phần lãnh thổ tối đa kể từ đường biên giới 2 vùng liền kề mà một tác tử nằm trong khu vực này của một vùng có thể " nhìn " được thông tin về trạng thái của hoặc " nghe " được thông điệp từ một tác tử vùng bên kia. Nói cách khác, cần xác định cơ chế để chương trình mô phỏng của một vùng có khả năng " nhìn/nghe " được những thay đổi trong khu vực giáp ranh của vùng liền kề của nó.

2. **Di cư của tác tử giữa hai vùng liền kề**: nếu tác tử nằm trong khu vực giáp ranh của một vùng quyết định di chuyển sang một điểm thuộc lãnh thổ của vùng liền kề thì cần có cơ chế cho phép " di cư tác tử liên mô phỏng ", nghĩa là tác tử sẽ được loại khỏi mô phỏng gốc (nhưng không phải " chết " hay "sống sót" như trong trường hợp với biên giới thật mà chỉ đơn thuần là tác tử bị " gạch tên " khỏi mô phỏng này), đồng thời nó được tái tạo trong mô phỏng đích tại điểm đến dự kiến (nhưng không tính là tác tử mới phát sinh mà danh sách tác tử của mô phỏng được tăng thêm một) với trạng thái như trong mô phỏng gốc.

3. **Đồng bộ hóa và hiển thị kết quả**: mặc dù các máy mô phỏng sẽ chạy cùng một chương trình và xử lý như nhau với từng loại tác tử, song vì các vùng sẽ không thể đồng đều về phân bố tác tử (số lượng và chủng loại), các máy mô phỏng về nguyên tắc là không thuần nhất (cả về cấu hình và tải) nên phải có cơ chế đảm bảo đồng bộ giữa các mô phỏng bộ phận: tất cả các mô phỏng đều phải chạy cùng một nhịp dựa trên bước mô phỏng. Ngoài ra, để việc song song hóa là trong suốt với người sử dụng thì kết quả của các mô phỏng bộ phận đều phải được tập trung sau mỗi bước mô phỏng tại một điểm để có thể được hiển thị cùng nhau. Nói cách khác, cần có cơ chế tách rời tính toán mô phỏng với hiển thị.

Trong mục tiếp theo, đề xuất một giải pháp cho các vấn đề nêu trên với trường hợp cụ thể là nền tảng mô phỏng GAMA sẽ được trình bày.

IV. GIẢI PHÁP TRÊN NỀN TẢNG MÔ PHỎNG GAMA

GAMA (*GIS Agent-based Modeling Architecture*) [1] là một nền tảng mô hình hóa và mô phỏng đa tác tử nguồn mở (giấy phép GPL v2) trên nền hệ thống thông tin địa lý GIS (*Geographical Information System*). GAMA hỗ trợ thiết kế các mô hình liên quan không gian địa lý, đa mô thức (*multi-paradigms*) và đa quy mô (*multi-scales*). Theo mô tả trên trang Wiki của dự án [3], kiến trúc của GAMA là một tổ hợp các *projects* Eclipse viết bằng Java, trong đó một số *projects* là là các thành phần lõi, không thể thiếu khi chạy các mô phỏng GAMA. Còn lại là các *plugins* có thể tùy biến và thêm/bớt theo yêu cầu của mô phỏng. Các mô phỏng đa tác tử được lập trình theo một ngôn ngữ riêng GAML [4].

Giải pháp song song hóa mô phỏng GAMA theo cách tiếp cận đã nêu trong mục trước sẽ không can thiệp vào lõi của GAMA mà chỉ gồm một số *plugins* và công cụ phụ trợ cần được phát triển thêm để giải quyết 3 vấn đề đã nêu và một quy trình để sửa đổi hành vi của trình điều khiển mô phỏng và các tác tử của ứng dụng mô phỏng nhằm phân biệt biên giới thông thường của không gian mô phỏng với biên giới ảo được thiết lập giữa các vùng liền kề. Cụ thể như sau:

1. **Công cụ hỗ trợ chia cắt không gian mô phỏng**: việc chia cắt không gian GIS của mô phỏng phụ thuộc vào đặc trưng riêng của mỗi bài toán mô phỏng. Không thể có một công thức cố định để tính xem mỗi mô phỏng sẽ được chia thành mấy vùng liền kề hay sử dụng cái gì (đường thẳng hay phần tử GIS) để làm đường chia cắt. Giải pháp cho việc chia cắt được đề xuất gồm 2 công cụ:

- một cấu trúc dữ liệu kiểu chuỗi (*list*) cho phép đặc tả đường biên giới ảo giữa hai vùng liền kề từ điểm đầu đến điểm cuối. Các phần tử của chuỗi thực chất là các phần tử GIS được đặc tả theo ngôn ngữ của GIS.
- một công cụ cho phép đánh dấu đường ranh giới giữa các vùng liền kề theo đặc tả đường biên giới ảo ở trên. Đường ranh giới này khác với đường biên giới của không gian mô phỏng và phải chứa thông tin cho biết số hiệu của vùng liền kề bên kia ranh giới.

2. **Công cụ hỗ trợ " nghe/nhìn " khu vực giáp ranh**: đây sẽ là một *plugin* mới của GAMA thực hiện các chức năng sau:

- hàm kiểm tra nếu tác tử nằm trong khu vực giáp ranh.
- thông báo cho mô phỏng liền kề về mọi thay đổi trạng thái của tác tử nằm trong khu vực giáp ranh hoặc thông điệp của tác tử dành cho tác tử phía vùng liền kề.

3. **Công cụ hỗ trợ " di cư tác tử liên mô phỏng "**: đây cũng là một *plugin* mới của GAMA thực hiện các chức năng sau:

- xóa tác tử quyết định di cư khỏi mô phỏng gốc
- chuyển toàn bộ trạng thái hiện thời của tác tử sang cho mô phỏng đích để tái tạo tác tử ở đó.

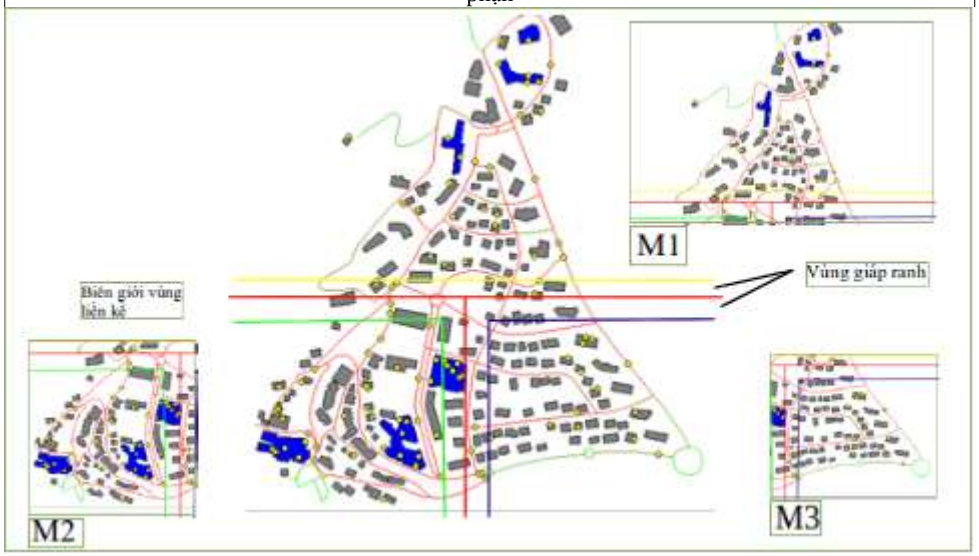
4. **Công cụ hỗ trợ đồng bộ hóa và hiển thị kết quả**: đây cũng là một *plugin* mới của GAMA thực hiện các chức năng sau:

- chuyển trạng thái mới của mọi tác tử có thay đổi của mỗi mô phỏng con về máy chủ hiển thị trung tâm.
- thông báo cho máy chủ hiển thi việc hoàn thành bước mô phỏng hiện thời
- thông báo thực hiện bước mô phỏng tiếp theo cho các mô phỏng con.

Để minh họa cho giải pháp, chúng tôi lấy ví dụ một mô phỏng có tên gọi " Road Traffic " (Giao thông đường phố) có trong kho mô phỏng giáo khoa (*tutorials*) của GAMA 1.7 (http://vps226121.ovh.net/tutorials#RoadTrafficModel)

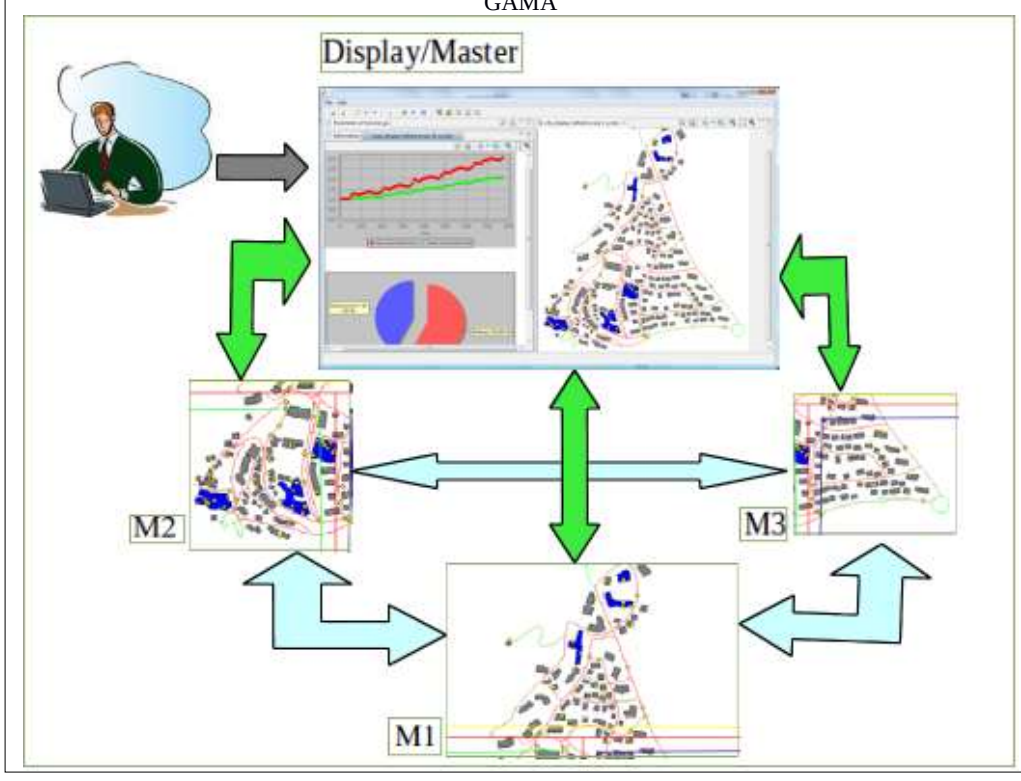
Hình 2 minh họa bản đồ môi trường được chia làm ba vùng liền kề với đường biên giới vùng và các khu vực giáp ranh. Trong ví dụ này, môi trường (bản đồ) của mô phỏng bộ phận M1sẽ bao gồm cả vùng giáp ranh bên phía M2 và M3 để giúp cho tác tử nằm trong lãnh thổ của M1 có thể " nghe/nhìn " những thay đổi xảy ra trong các khu vực giáp ranh này. Tương tự với M2 và M3.

Hình 2. Chia mô phỏng GAMA thành các mô phỏng bộ phận



Hình 3 minh họa việc triển khai 3 mô phỏng bộ phận chạy song song trên ba máy M1, M2 và M3. Một máy Display/Master đóng vai trò hiển thị và giao tiếp với người sử dụng. Các mô phỏng bộ phận đều được thực hiện theo kiểu không hiển thị kết quả (*headerless*) mà chỉ thông báo mọi thay đổi trong vùng của mình về cho máy chủ Display/Master là nơi lưu giữ kết quả mô phỏng toàn bộ. Máy chủ này chỉ làm nhiệm vụ hiển thị kết quả cho người sử dụng trên cơ sở trạng thái của các tác tử nhận được từ các mô phỏng bộ phận.

Hình 3. Giải pháp song song hóa mô phỏng trên nền tảng GAMA



Để thực hiện được mô hình giải pháp này, các tác tử trong các ứng dụng mô phỏng cũng phải được thay đổi trong hành vi. Nói một cách khác, ngoài việc phát triển các công cụ và plugins bổ sung cho nền tảng GAMA như đã đề cập ở trên, chúng ta sẽ còn phải " song song hóa " các tác tử cho mỗi ứng dụng mô phỏng có nhu cầu chạy song song. Vì số chủng loại tác tử trong mỗi mô phỏng thường có số lượng nhỏ nên lập trình viên mô phỏng có thể dễ dàng thích ứng một mô phỏng đã được lập cho một máy thành phiên bản " song song " trong một thời gian tương đối ngắn.

Quy trình " song song hóa " một ứng dụng mô phỏng GAMA sẽ quy về việc áp dụng thay đổi với mọi tác tử Ai của ứng dụng theo " thuật toán " sau:

```
S0 ← Status(Ai)
S1 ← Status(Process(Ai))
If (S1 # S0) then
      Inform (Ai, S1, Master)
      If (Ai ∈ Border_area (Mj)) then
            Inform (Ai, S1, Mj)
      If ( Ai goto Mj) then
            Create (Ai, S1, Mj)
            Clear (Ai)
```

Ở đây `Status(Ai)` được hiểu là tập hợp các thuộc tính của tác tử có thể bị thay đổi khi áp dụng xử lý hành vi của Ai (`Process(Ai)`). Các hàm `Inform(), Create()` và `Clear()` là các API mới do các plugins đề cập ở điểm 3 ở trên cung cấp. Tham số của các API chỉ mang tính minh họa.

Vấn đề tách rời thực hiện mô phỏng thuần túy (*headerless*) với hiển thị kết quả mô phỏng đã được trù tính trong GAMA nên việc chuyển các mô phỏng thành phần sang dạng headerless sẽ không gặp khó khăn. Các thuật toán xử lý hành vi của tác tử trong mỗi bước mô phỏng không bị thay đổi.

Ngoài những thay đổi áp dụng cho các tác tử đã mô tả ở trên, chúng ta sẽ còn phải áp dụng một thay đổi nữa cho lớp điều khiển mô phỏng liên quan đến việc thực hiện bước tiếp theo tại các mô phỏng thành phần và máy chủ Master. Chương trình mô phỏng thành phần sẽ không được chạy " hết tốc lực " mà phải chờ đợi tín hiệu đồng bộ do máy chủ Master phát ra. Master sẽ chỉ phát tín hiệu bước mô phỏng tiếp theo sau khi đã nhận được thông báo hoàn thành bước mô phỏng hiện thời của tất cả các mô phỏng thành phần.

V. CÀI ĐẶT VÀ ĐÁNH GIÁ

Với những phân tích tổng quát và giải pháp đề xuất trong bài báo này, việc thiết kế chi tiết và cài đặt giải pháp có thể bắt đầu. Hiện nay công việc này đang được tiến hành trên nền tảng GAMA 1.7 tại phòng nghiên cứu MSI của Viện quốc tế Pháp ngữ, ĐGQGHN. Chúng tôi hy vọng đến cuối 2016 sẽ hoàn thành phiên bản đầu cho giải pháp song song hóa và kết quả thực nghiệm trên một vài mô phỏng ví dụ, qua đó có thể khẳng định tính khả thi của tiếp cận và đánh giá được hiệu quả của giải pháp.

Thật vậy, dễ thấy cách tiếp cận sẽ gặp một số hạn chế như lượng thông tin trao đổi liên mô phỏng sẽ là đáng kể và là một cản trở lớn cho hiệu quả của giải pháp. Cơ chế đồng bộ bắt buộc cũng sẽ là một nhân tố tăng thêm *overhead* cho giải pháp. Mô hình máy chủ hiển thị tập trung tiềm ẩn nguy cơ " thắt nút cổ chai " khi vận hành thực tế.

Để khắc phục những hạn chế này, các giao tiếp liên mô phỏng sẽ cần được chú ý và tối ưu hóa ngay từ đầu trong thiết kế và lập trình các plugins và API mới. Một trong những giải pháp tối ưu thiết kế là giảm thiểu tối đa lượng thông điệp liên mô phỏng bằng cách gộp các thông tin trạng thái vùng giáp ranh vào một thông điệp chung để gửi đến mô phỏng liền kề sau mỗi bước mô phỏng. Tương tự với các thông tin thay đổi trạng thái của các mô phỏng thành phần gửi đến máy chủ hiển thị kết quả. Cần tận dụng tối đa các API đã có để tiết kiệm công sức và tránh mâu thuẫn thiết kế.

Những thông số cần được đo lường và đánh giá trên prototype sẽ bao gồm overhead của giao thức trao đổi thông tin và của giao thức đồng bộ hóa, hiển thị tập trung.

VI. KẾT LUẬN VA TRIỂN VỌNG

Trong bài báo này, chúng tôi đã trình bày chi tiết một phương pháp tiếp cận mới nhằm cải thiện năng lực cho các hệ nền nền tảng mô hình hóa và mô phỏng đa tác tử trên nền hệ thống thông tin địa lý, dành cho những mô phỏng với số lượng rất lớn tác tử mà một máy PC cấu hình mạnh cũng sẽ không kham nổi. Sau khi phân tích những đặc thù của loại hình mô phỏng này, chúng tôi đề xuất một cách tiếp cận cũng theo tư tưởng " chia để trị ", nhưng việc chia được dựa trên đặc thù môi trường GIS mà các tác tử được triển khai để thực hiện mô phỏng. Môi trường mô phỏng ban đầu sẽ được chia thành các vùng liền kề và mỗi vùng sẽ được nạp cho một máy PC chạy song song cùng một chương trình mô phỏng nhưng với môi trường và số lượng tác tử thu nhỏ hơn rất nhiều so với mô phỏng gốc. Với cách tiếp cận này, phần lõi của hệ nền sẽ không phải thay đổi và mở ra khả năng mở rộng qui mô mô phỏng một cách tuyến tính với số lượng tác tử và môi trường địa lý.

Một giải pháp cụ thể cho nền tảng GAMA theo hướng tiếp cận này đã được mô tả chi tiết, đủ sẵn sàng cho thiết

kế và cài đặt thử nghiệm. Giải pháp không yêu cầu thay đổi cấu trúc và thuật toán của GAMA. Những công cụ và API (*plugins*) bổ sung cho GAMA được đề xuất và đặc tả chi tiết. Một quy trình dễ hiểu để "song song hóa" một ứng dụng mô phỏng trên GAMA cũng được đặc tả đầy đủ.

Công việc thiết kế và cài đặt thử nghiệm hiện mới được bắt đầu nên chúng tôi còn chưa thể đưa ra các đánh giá khách quan có số liệu minh họa về tính khả thi và hiệu quả của cách tiếp cận.

VII. LỜI CẢM ƠN

Công trình nghiên cứu này được tài trợ bởi đề tài nghiên cứu khoa học cấp ĐHQGHN "Phương pháp và công cụ mô phỏng ứng dụng trong việc tổ chức cứu hộ khi có sự cố cháy tại các địa điểm đông người (siêu thị, trung tâm mua sắm)", mã số QG.15.31.

TÀI LIỆU THAM KHẢO

[1] A. Grignard, P. Taillandier, B. Gaudou, D-A. Vo, N-Q. Huynh, A. Drogoul (2013), GAMA 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation. In 'PRIMA 2013: Principles and Practice of Multi-Agent Systems', Lecture Notes in Computer Science, Vol. 8291, Springer, pp. 117-131. http://link.springer.com/chapter/10.1007/978-3-642-44927-7_9.

[2] S. Bhamidipati (edited), A subset of the scientific papers that have been written either about GAMA or using the platform as an experimental/modeling support. https://github.com/gama-platform/gama/wiki/References.

[3] L. Mazars (edited), Architecture of GAMA. https://github.com/gama-platform/gama/wiki/GamaArchitecture.

[4] GAMA Project, GAML References. http://vps226121.ovh.net/references#GamlReference.

[5] Fuyuki Shimojo, Rajiv K. Kalia, Aiichiro Nakano and Priya Vashishta. Embedded divide-and-conquer algorithm on hierarchical real-space grids: parallel molecular dynamics simulation based on linear-scaling density functional theory. In Computer Physics Communications 167 (2005) 151–164.

[6] Anne Hakansson and Ronald Hartung. Autonomously creating a hierarchy of intelligent agents using clustering in a multi-agent system. In Proceedings of the 2008 International Conference on Artificial Intelligence, ICAI 2008, July 14-17, 2008, Las Vegas, Nevada, USA. Available on https://www.researchgate.net/publication/220834565.

[7] Aiichiro Nakano, Rajiv K. Kalia, Ken-ichi Nomura, Ashish Sharma, Priya Vashishta, Fuyuki Shimojo, Adri C.T. van Duin, William A. Goddard, Rupak Biswas and Deepak Srivastava. A divide-and-conquer/cellular-decomposition framework for million-to-billion atom simulations of chemical reactions. In Computational Materials Science 38 (2007) 642–652.

PARALLEL PROCESSING GIS-BASED MULTI-AGENTS SIMULATIONS BY DIVISION IN PIECES

Nguyễn Hồng Quang, Hồ Tường Vinh, Nguyễn Mạnh Hùng

ABSTRACT*— The multi-agent simulation tools on geographical information environment (Geographical Information System - GIS) as GAMA (GIS Agent-based Modeling Architecture) often allow a simulation running on only one server. With the huge simulation for complex systems with large numbers of agents (hundreds of thousands or millions), the conventional PC will not be able to process or processing time will be excessive . The parallelization schemes of simulation tools in order to speed up the host processor on multi-core parallel server give limited results and require review the architecture and design of simulation tools, simulation programs and accompanying language.*

We propose a comprehensible approach independent with tools and language simulation, based on geographic information environment aspect: splitting simulation geographical environment into small pieces (sub-regions) together with the agents initially deployed on its, then use multiple servers running in parallel with a same simulation program, but each in charge of one sub-region. Synthesis of these sub-simulations with their mutual relationship will give the general simulation results. In this paper, we first analyze and list the possible relationship between agents on adjacent regions. Then we propose solution to address the problems caused by these relationships on the simulation platform GAMA. A simulation application implementing this approach on GAMA will prove the feasibility of the proposed approach and point out the problems that need solving for the real application of the approach.

Keywords *— Parallel simulations, multi-agent, GIS, GAMA.*