KHÁM PHÁ TÀI NGUYÊN LƯỚI HƯỚNG NGƯỜI DÙNG

USER-DRIVEN GRID RESOURCE DISCOVERY

Nguyễn Quang Hùng, Nguyễn Thanh Sơn
Khoa Khoa Học & Kỹ Thuật Máy Tính
Nhà A3, Trường Đại học Bách Khoa – ĐHQG Tp.HCM
hungnq2@cse.hcmut.edu.vn, sonsys@hcmut.edu.vn

BẢN TÓM TẮT

Tính toán lưới (grid computing) là một nền tảng tính toán mới và GT (Globus Toolkit) là một môi trường hiện thực nền tảng tính toán lưới phổ dụng hiện thời. Tuy nhiên, thành phần MDS (Monitoring and Discovery System) của GT chưa có một dịch vụ tìm kiếm tài nguyên tốt. Bài báo này trình bày một phương pháp khả thi cho phép tìm kiếm tài nguyên trên lưới theo nhu cầu của người sử dụng. Người dùng có thể đặc tả yêu cầu tài nguyên dưới dạng mở rộng của chuẩn JSDL (Job Specification Description Language) – trong đó mỗi tài nguyên được gán một khoảng trị có cận dưới và cận trên. Giải pháp này được phát triển dưới dạng một dịch vụ lưới chuẩn WSRF (Web Service Resource Framework), đa luồng viết bằng Java. ẩ goài ra, bài báo còn trình bày giải thuật so trùng nhóm (set-matching) và xếp hạng các tài nguyên.

ABSTRACT

Grid computing is a new computing infrastructure and Globus Toolkit (GT) is a de-factor grid middleware to build grid. However, the Globus Toolkit's Monitoring and Discovery System (MDS) do not support a resource discovery service well. So this paper presents a user-driven grid resource discovery service. Users can describe resource requirement specifications by extended Job Specification Description Language (JSDL), in which each resource is given a range value with a lower bound and an upper bound. Our discovery service is Web Service Resource Framework (WSRF), Java multi-thread grid service. In addition, the paper introduces the resource set-matching and ranking algorithm.

1. Giới thiệu

Lưới tính toán bao gồm nhiều nút lưới chia xẻ và cộng tác [9], [10] với nhau theo các chuẩn (hay giao thức) đang được xác lập như Open Grid Services Architecture (OGSA) [12] và Web Service Resource Framework (WSRF) [5]. Mỗi nút lưới có thể là một máy tính cụm (PC-based cluster) hay một máy trạm (workstation). Số lượng tài nguyên tham gia trong lưới thay đổi nhanh và người sử dụng có thể chưa biết tất cả tài nguyên hiện hữu trong lưới tính toán. Vì vậy một bài toán đặt ra là làm sao để người sử dụng tìm được các tài nguyên có trên lưới thỏa mãn các ràng buộc của mình.

Đây là một vấn đề khó đang được nghiên cứu. ẩ guyên nhân là do số lượng người dùng trên lưới là lớn và họ có nhu cầu thực thi các công việc khác nhau với các ràng buộc khác nhau. Ví dụ có công việc cần tài nguyên là bộ xử lý mạnh nhưng công việc khác lại quan tâm và đòi hỏi tài nguyên có bộ nhớ lớn; hay người dùng có thể đưa ra các ràng buộc trên cả số lượng bộ xử lý cho mỗi máy và dung lượng trống còn lại của bộ nhớ cũng như số lượng tổng số bộ xử lý cần… Hơn nữa, trong việc khám phá tài nguyên thỏa mãn các ràng buộc phải làm thế nào để các tài nguyên tìm được thỏa mãn yêu cầu đặt ra nhưng không được vi phạm các yêu cầu tối thiểu. Ví dụ người dùng yêu cầu tài nguyên X có dung lượng bộ nhớ thích hợp là lớn hơn hay bằng 512 Mbytes (nhưng tối thiểu phải là 128 Mbytes), và như vậy trong kết quả tìm kiếm thì các tài nguyên tính toán có $X \geq 512$ Mbytes được xếp trước các tài nguyên có 128 Mbytes $\leq X \leq 512$ Mbytes. Trong

bài báo này, công việc lựa chọn tài nguyên, xếp hạng tài nguyên và gom tài nguyên vào kết quả cuối cùng trong quy trình khám phá tài nguyên trên lưới đã được nghiên cứu, hiện thực và thử nghiệm.

Giải pháp khám phá tài nguyên trên lưới được hiện thực dưới dạng một dịch vụ lưới tên gọi VOResDiscovery tương thích chuẩn OGSA [12] và WSRF [5], và nó có thể được cài vào môi trường Globus Toolkit 4.0.x (GT4) (hay phiên bản tương thích). Đồng thời để dễ dàng hơn cho người dùng và ứng dụng đầu cuối, một shell script và Java API đã được cung cấp để giao tiếp với dịch vụ VOResDiscovery. ầ ếu dùng shell script eda_resource_discovery thì người dùng chỉ cần chạy script trên môi trường "bash shell" và tham số là đường dẫn chỉ đến tập tin mô tả công việc và ràng buộc trên các tài nguyên cần tìm (Job Description) [1]. Còn nếu dùng Java API thì lớp chính VORDSClient có phương thức discovery(String strXmlJobDesc, int nMax) sẽ gửi yêu cầu đến dịch vụ VOResDiscovery.

Đặc điểm nổi bật của giải pháp này là người dùng có thể khám phá ra tài nguyên theo các ràng buộc riêng trên từng loại công việc trước khi gửi đến hệ thống tính toán lưới. Hiện tại thành phần MDS-4 [8], [11], [13] của GT4 và các grid middleware khác như Condor [2]/Condor-G [3] vẫn chưa cung cấp, và người dùng sẽ không phải lựa chọn bằng tay các tài nguyên mà những việc này đã được đơn giản hóa như một shell script hay bằng một lời gọi phương thức Java.

Các phần còn lại của bài báo được bố cục như sau: phần 2 sẽ giới thiệu các công việc liên quan, phần 3 sẽ giới thiệu về mô hình dịch vụ khám phá tài nguyên lưới hướng người dùng. Kết quả thử nghiệm dịch vụ về tốc độ và mức độ ổn định của 1000 yêu cầu tuần tự gửi đến được trình bày ở phần 4. Phần cuối cùng là kết luận và hướng phát triển trong tương lai với dịch vụ khám phá tài nguyên này.

2. Các công việc liên quan

Trong GT4, thành phần MDS-4 [8], [11], [13] sẽ quản lý thông tin của các tài nguyên và dịch vụ WSRF [5] dưới dạng một phân cấp. Thông tin được gửi từ các nguồn khác nhau và tổng hợp lại bằng dịch vụ Index của MDS-4 (MDS-4 có ba dịch vụ mức cao là dịch vụ Index, Trigger và Aggregation). ả hưng dịch vụ Index này không hỗ trợ đặc tả việc tìm tài nguyên tính toán có nhiều ràng buộc đồng thời trên các yếu tố về bộ xử lý, bộ nhớ, dung lượng đĩa cứng, môi trường hệ điều hành, chứng chỉ người dùng…

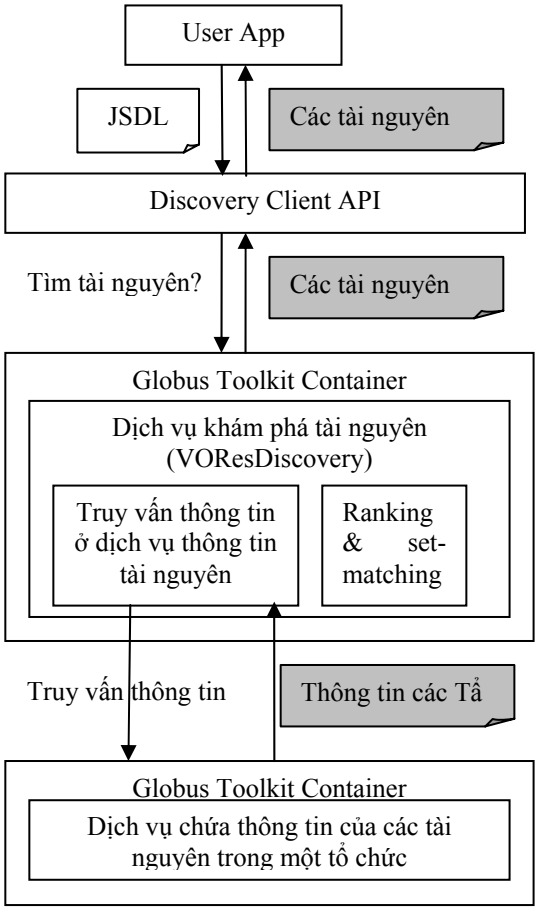
Condor [2] là một middleware cho phép gom các máy trạm riêng lẻ lại với nhau. Condor Matchmaker là một thành phần dùng để khám phá ra các máy tính trạm riêng lẻ thỏa mãn các ràng buộc bằng ngôn ngữ ClassAd. Tuy nhiên việc khám phá tài nguyên trên lưới hoàn toàn khác và Condor-G [3] hiện thiếu phần này.

Một số xu hướng khác dùng các công nghệ của web ngữ nghĩa để khám phá tài nguyên trên môi trường tính toán lưới như trong [19]. Tuy nhiên các tác giả trong [19] vẫn không giải quyết được vấn đề tìm tập các tài nguyên tính toán (các nút lưới) có tổng số bộ xử lý theo mong muốn của người dùng như được nêu ra ở phần giới thiệu.

3. Mô hình dịch vụ khám phá tài nguyên hướng người dùng

3.1. Mô hình dịch vụ khám phá tài nguyên

Mô hình dịch vụ khám phá tài nguyên (Hình 1) này được thiết kế bao gồm: (i) một thư viện API phía client và (ii) dịch vụ khám phá tài nguyên trên lưới (chuẩn WSRF [5]). Phía người dùng, các ứng dụng có thể tạo ra đối tượng *VORDSClient*, rồi gọi phương thức *discovery (String xmlJsdl, int nMax)* của VORDSClient. Thư viện phía client này sẽ giúp việc lập trình tương tác phía người dùng dễ hơn với dịch vụ khám phá tài nguyên, nghĩa là người dùng chỉ cần quan tâm đến việc tạo ra một chuỗi XML mô tả các tài nguyên cần để chạy công việc (trên môi trường tính toán lưới) theo định dạng JSDL mở rộng [1]. Kết quả trả về của phương thức discovery() phía client là một chuỗi XML kết quả tìm kiếm [15].



Hình 1: Mô hình dịch vụ khám phá tài nguyên

Bên cạnh đó, người dùng có thể gọi một shell script để gửi một yêu cầu tìm kiếm tài nguyên đến dịch vụ khám phá tài nguyên hoặc gọi trực tiếp bằng giao thức WSRF [5].

Dịch vụ khám phá tài nguyên sẽ nhận yêu cầu gửi từ phía người dùng, truy vấn thông tin các tài nguyên tính toán lưới từ dịch vụ thông tin, lọc ra các tài nguyên thích hợp dựa trên yêu cầu của người dùng, xếp hạng tài nguyên và gửi trả kết quả về cho phía client theo định dạng XML <RDSResult> … <RDSResult> [15]. Giải thuật lọc, xếp hạng (ranking) và so trùng nhóm (set-matching) tài nguyên sẽ được trình bày trong phần 3.3 và 3.4.

3.2. Định dạng mô tả yêu cầu của người dùng (Job Description)

ả gười dùng sẽ mô tả tài nguyên cần để thực thi một công việc trên môi trường tính toán lưới theo định dạng được quy định trong tài liệu [1]. Ở đây chỉ nêu ra một ví dụ minh họa yêu cầu tìm kiếm tài nguyên của người dùng. Giả sử người dùng cần chạy một ứng dụng MPI trên môi trường tính toán lưới với điều kiện: tổng số bộ xử lý mong muốn là 20 (nhưng tối thiểu từ 1), đồng thời trên mỗi máy phải thỏa mãn các yêu cầu riêng (thẻ <Individual*>) gồm: số bộ xử lý mong muốn là 2 (tối thiểu là 1), băng thông của mạng là 80 Mbps (tối thiểu là 5 Mbps), bộ nhớ RAM là 1024 Mbytes (tối thiểu là 500 Mbytes) và sức mạnh của mỗi bộ xử lý là 2000 MFlops (tối thiểu là 100 MFlops); và về phía ứng dụng MPI thì môi trường thực thi có Platform là i686-*-Linux hoặc SunOS và tiêu đề của chứng chỉ người dùng trên lưới là "/O=Grid/OU=GlobusTest/OU=simpleCA-sunfiren0.edagrid.hcmut.edu.vn/OU=edagrid.hcmut.edu.vn/Cả =edagrid". Bảng 1 bên dưới sẽ mô tả yêu cầu này dưới dạng chuỗi XML như trong tài liệu [1].

Bảng 1: Ví dụ về một mô tả công việc từ phía người dùng để khám phá tài nguyên cho ứng dụng

```
<?xml version="1.0" encoding="UTF-8" ?>
<JobDefinition> <JobDescription>
<Resources>
<TotalCPUCount lowerbound="1"> 20
</TotalCPUCount>
<IndividualCPUCount lowerbound="1"> 2
</IndividualCPUCount>
<ả etworkBandwidth lowerbound="5"> 80
</ả etworkBandwidth>
<IndividualPhysicalMemoryFree
        lowerbound="500"> 1024
</IndividualPhysicalMemoryFree>
<IndividualCPUPower lowerbound="100"> 2000
</IndividualCPUPower>
</Resources>
<Application type="MPI">
<Executable Platform= "i686-*-Linux">
http://www.cse.hcmut.edu.vn/~user/myprog.linux
</Executable>
<Executable Platform="sparc-Sun-SunO" >
http://www.cse.hcmut.edu.vn/~user/myprog.sunos
</Executable>
<Argument>-input</Argument>
<Argument>file1</Argument>
<Argument>-output</Argument>
<Argument>file2</Argument>
</Application>
<User>/O=Grid/OU=GlobusTest/OU=simpleCA-
sunfiren0.edagrid.hcmut.edu.vn/
OU=edagrid.hcmut.edu.vn/Cả =edagrid
</User>
</JobDescription> </JobDefinition>
```

3.3. Giải thuật khám phá tài nguyên trên lưới

Giải thuật khám phá tài nguyên bao gồm các bước chính sau:

Bước 1: Dịch vụ VOResDiscovery nhận yêu cầu từ phía người dùng.

Bước 2: Phân tích yêu cầu này.

Bước 3: Chuyển đổi thành các câu truy vấn thích hợp và gửi đến các dịch vụ thông tin (quản lý tài nguyên lưới).

Bước 4: ả hận kết quả từ dịch vụ thông tin tài nguyên lưới.

Bước 5: Phân tích thông tin các tài nguyên lưới được trả về.

Bước 6: Thực hiện việc loại bỏ các tài nguyên lưới do không còn bộ xử lý rảnh, hoặc không thỏa mãn các yêu cầu tối thiểu riêng trên từng thẻ <Individual*> trong tài liệu [1] như là <IndividualCPUCount />, hay <IndividualPhysicalMemoryFree />, hoặc thiếu thông tin về URL của các dịch vụ cần thiết (xác định cụ thể cho từng hệ thống lưới)…

Bước 7: Trên tập tài nguyên còn lại (tất cả đều thỏa mãn yêu cầu tối thiểu riêng trên từng thẻ <Individual*>), gọi là R_{ind}, ta thực việc giải thuật ranking (trình bày ở phần 3.4) và giải thuật so trùng nhóm (set-matching) để lấy ra kết quả bao gồm tập các tài nguyên có thứ tự (đã xếp hạng) nhằm thỏa mãn các ràng buộc tổng cộng (như thẻ <TotalCPUCount> [1]).

Bước 8: Tạo ra chuỗi XML mô tả kết quả khám phá tài nguyên và gửi về cho phía người dùng.

Giải thuật set-matching được thực hiện trên cơ sở:

i. Gom tất cả các tài nguyên lưới đầu tiên (xếp hạng cao nhất) thỏa các ràng buộc cận trên trước, gọi là $R_{selected}$.
ii. Việc thu gom đến khi thỏa mãn các yêu cầu tổng cộng trên tập tài nguyên được chọn $R_{selected}$, hoặc khi số lượng tài nguyên của tập $R_{selected}$ bằng số lượng tối đa mà người dùng muốn nhận (ví dụ người dùng chọn tối đa là hai mươi tài nguyên trong tổng số triệu tài nguyên trên toàn môi trường tính toán lưới).

3.4. Giải thuật xếp hạng (ranking) tài nguyên theo yêu cầu của công việc

Giải thuật xếp hạng tài nguyên sẽ dựa trên mối quan hệ giữa ràng buộc về yêu cầu tài nguyên riêng của công việc (các thẻ <Individual* />) và các tài nguyên trên các nút lưới. Các thẻ <Total* /> được giải quyết với giải thuật so trùng nhóm (set-matching). Đầu vào của giải thuật xếp hạng là tập tài nguyên R_{ind} (R_{ind} là kết quả ở bước 6 trong giải thuật khám phá tài nguyên, phần 3.3).

Bảng 2: Giải thuật xếp hạng tài nguyên

```
FUả CTIOả  Ranking_Algorithm
FOR each (tài nguyên Ri thuộc tập Rind )
   T = 0;
   FOR each (Attr(Job, j) là thuộc tính j của Job)
      T = T + [w(Attr(Job,j)) * Attr(Ri,j)/Attr(Job,j)];
   ả ext thuộc tính kế tiếp
   Rank(Ri) = T;
ả ext tài nguyên kế tiếp

Sắp xếp tập các tài nguyên Rind theo Rank(Ri) từ cao
nhất đến thấp nhất

Return Danh sách tài nguyên đã sắp xếp thứ tự
```

Chú giải các ký hiệu:

- R_{ind} là tập tài nguyên đã được loại bỏ các tài nguyên không thỏa mãn ở bước 6 của giải thuật khám phá tài nguyên (phần 3.3).
- Attr(Job, j) là thuộc tính j của công việc Job
- Attr(R_i, j) là thuộc tính j của tài nguyên R_i
- w(Attr(Job,j)) là trọng số của thuộc tính j của công việc.
- Rank(R_i) là điểm số của tài nguyên R_i được dùng để xếp hạng tài nguyên.

Giả sử n số tài nguyên trong tập R_{ind} , gọi m là số thuộc tính có trong Job thì độ phức tạp về thời gian của giải thuật xếp hạng trên sẽ là O(m*n).

3.5. Truy vấn thông tin từ dịch vụ thông tin trên của lưới

Dịch vụ VOResDiscovery sẽ dùng lớp QueryResourceInfo để giao tiếp với dịch vụ thông tin của lưới (như là dịch vụ thông tin tài nguyên lưới- VOInfo [7]). Tương lai chúng tôi sẽ phát triển tiếp để cho phép truy vấn và dùng được thông tin trên nhiều dịch vụ thông tin khác.

4. Thử nghiệm và đo đạc

4.1. Cách thức đo đạc

Phép đo đạc được thực hiện ở cả hai phía Client và phía dịch vụ khám phá tài nguyên VOResDiscovery. Viết một chương trình tuần tự dùng vòng lặp thực hiện ả lần việc tạo ra ả đối tượng VORDSClient, rồi lấy mốc thời gian trước và sau các phát biểu gọi phương thức discovery() của chúng. Trong phần đo đạc thực nghiệm chúng tôi cho ả = 1000.

Phía Client: ta thực hiện ghi nhận thời điểm trước và sau khi gọi tác vụ khám phá tài nguyên (discovery) từ xa. Khoảng thời gian này sẽ bao gồm: (1) thời gian client gửi và nhận các thông điệp SOAP đến/từ dịch vụ và (2) thời gian dịch vụ VOResDiscovery xử lý phía dịch vụ.

Phía dịch vụ khám phá tài nguyên VOResDiscovery: ta thực hiện ghi nhận thời điểm bắt đầu và kết thúc tác vụ discovery(). Khoảng thời gian xử lý yêu cầu phía dịch vụ sẽ gồm thời gian truy vấn thông tin từ dịch vụ VOInfo và thời gian tính toán cục bộ. Trên biểu đồ chỉ thể hiện thời gian truy vấn dịch vụ VOInfo – thời gian gọi thủ tục queryVOInfo và thời gian xử lý trọn vẹn một yêu cầu ở phía dịch vụ – thời gian gọi thủ tục discovery (không phân chia nhỏ cho từng phương thức như tạo truy vấn, so trùng dữ liệu, xếp hạng, chuyển thành chuỗi XML trả về, hợp lại và xóa các Java thread xử lý đã tạo ra).

Môi trường đo đạc: hai máy gồm Gridnode10 và EDAGrid server. Trong đó máy Gridnode10 chạy chương trình thử nghiệm TestVORDSClient, còn máy EDAGrid server đóng vai trò VO server (hosting và thực thi các dịch vụ như VOResDiscovery, VOInfoService...).

Cấu hình các máy như sau: máy Gridnode10 là một máy tính cá nhân, bộ xử lý Intel ® Pentium ® IV 3.0 GHz hỗ trợ HyperThreading, bộ nhớ là 1024 MBytes, đĩa cứng có dung lượng 80 Gbytes, chạy hệ điều hành RedHat Workstation phiên bản 4.0 với kernel 2.6, GT 4.0.3 và Postgres SQL 7. Cấu hình của máy EDAGrid server (VO Server) là máy chủ hiệu IBM xSeries 206, bộ xử lý Intel ® Pentium ® IV tốc độ 3.0 GHz hỗ trợ HyperThreading, bộ nhớ là 2048 Mbytes, đĩa cứng có dung lượng 320Gbytes, chạy hệ điều hành Debian 1.3.3.6-10 với Linux kernel 2.4.27-2-686, GT 4.0.3 và MySQL 4.5.

4.2. Kết quả tổng hợp sau khi đo đạc

Việc đo đạc thực hiện với 1000 yêu cầu được gửi tuần tự từ phía client (VORDSClient) đến dịch vụ khám phá tài nguyên (VOResDiscovery) rồi nhận kết quả. Các thuật ngữ dùng trong kết quả đo đạc:

Time_Client: Thời gian đo ở phía client tiêu tốn cho mỗi yêu cầu khám phá tài nguyên. Thời gian này được tính từ lúc bắt đầu tạo đối tượng VORDSClient đến lúc nhận được kết quả.

Time_Client =

Start_Time_Client – Finish_Time_Client (1).

Time_VOResDiscoveryService_Processing: Thời gian xử lý của dịch vụ VOResDiscovery cho mỗi yêu cầu khám phá. Bao gồm cả thời gian truy vấn thông tin tài nguyên từ dịch vụ VOInfo.

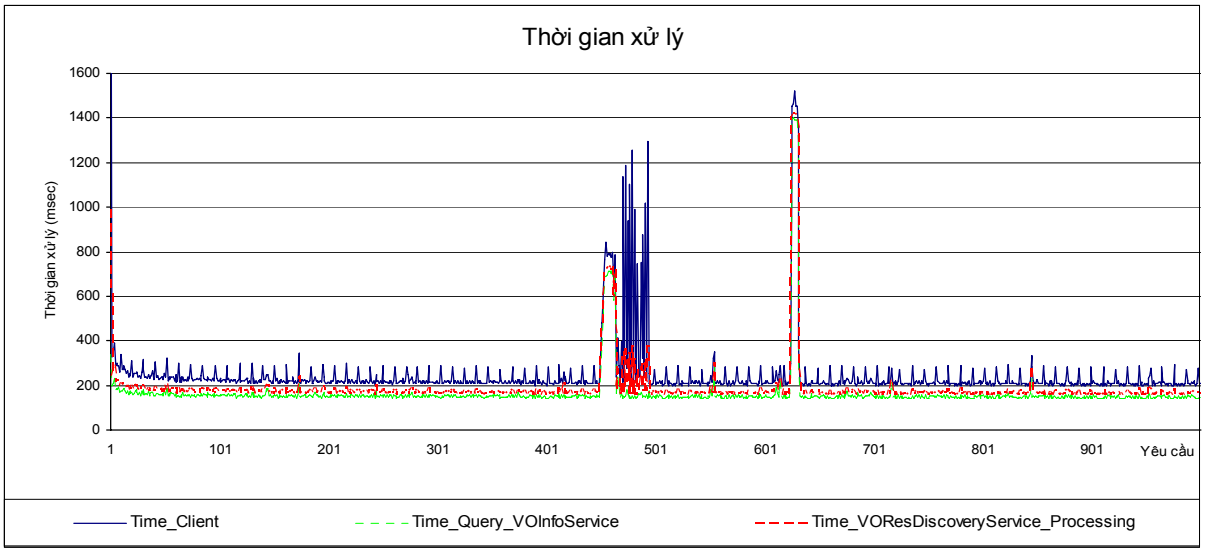
Time_VOResDiscoveryService_Processing =

Start_Discovery_Time – Finish_Discovery_Time (2)

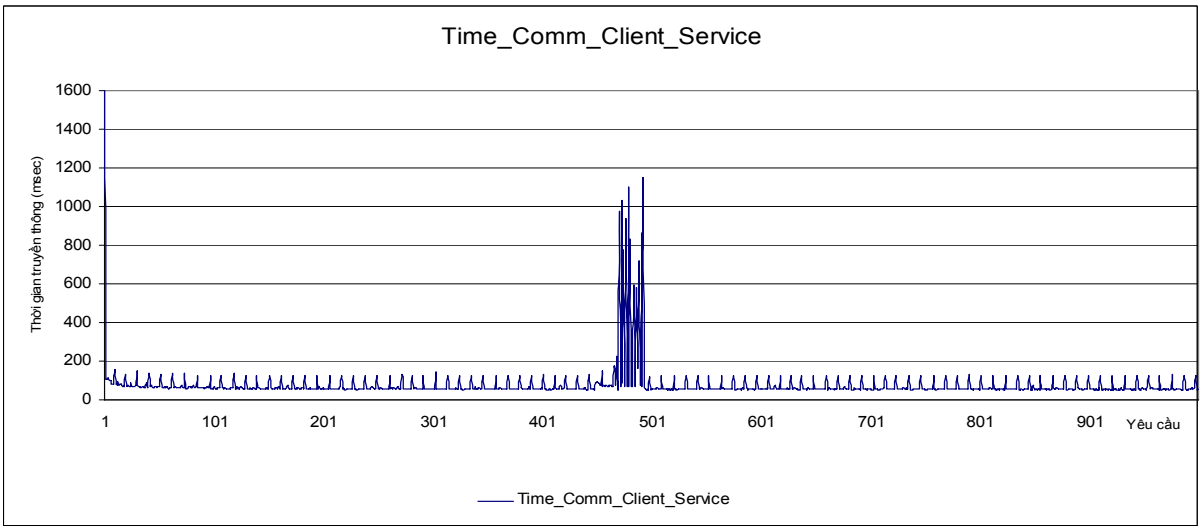
Time_Query_VOInfo: Thời gian truy vấn thông tin tài nguyên (gridnode) từ dịch vụ thông tin VOInfo.

Time_Query_VOInfo =

Start_Query_Time – Finish_Query_Time (3).



Hình 2: Biểu đồ thời gian cho 1000 yêu cầu tuần tự. Đường nét liền màu xanh da trời (Time_Client) là thời gian đo được tại phía client cho mỗi yêu cầu. Đường nét đứt dài màu đỏ (Time_VOResDiscovery_Processing) là thời gian xử lý tổng cộng tại phía dịch vụ khám phá tài nguyên VOResDiscovery (bao gồm thời gian truy vấn dịch vụ thông tin VOInfo – đường nét đứt ngắn liên tục màu xanh lá cây là Time_Query_VOInfo)



Hình 3: Tổng hai thời gian gửi thông điệp SOAP từ client đến server để gọi dịch vụ và thời gian gửi kết quả từ dịch vụ về client (Time_Client_Service).

Bảng 3: Thời gian trung bình (1000 yêu cầu) cho: (1) phía client, (2) phía dịch vụ khám phá tài nguyên, (3) truy vấn dịch vụ thông tin, và (4) thời gian gửi nhận thông điệp SOAP của client và server

Thời gian	Trung bình cho 1000 yêu cầu (mili-giây)
Time_Client (1)	250,25
Time_VOResDiscoveryService_ Processing (2)	174,30
Time_Query_VOInfo (3)	167,47
Time_Comm_Client_Service (4)	75,95

Time_Comm_Client_Service: Tổng hai thời gian gửi thông điệp SOAP từ client đến server để gọi dịch vụ và thời gian gửi kết quả từ dịch vụ về client.

Time_Comm_Client_Service = (1) – (2) (4).

Bảng 3 trình bày kết quả về thời gian trung bình cho 1000 yêu cầu của các tham số (1), (2), (3) và (4).

Hình 2 và Hình 3 cho thấy tại các yêu cầu từ khoảng 480 đến 500 và khoảng 625 đến 632 xuất hiện các điểm tăng đột biến. ẩ guyên nhân của việc tăng này do: trong khoảng 480 đến 500 các đồ thị cho thấy: (1) hệ thống ở phía server xử lý quá lâu (thời gian truy vấn dịch vụ VOInfo lớn) nên dẫn đến phía client cũng sẽ chờ nhận được kết quả (khám phá tài nguyên) lâu và (2) sau đó do phía client và server bị tắt nghẽn ở việc gửi thông điệp SOAP cho nhau, nên thời gian xử lý yêu cầu trong trường hợp này nhỏ nhưng thời gian truyền thông lại lớn; và trong

khoảng từ 625 đến 632 thì thời gian truy vấn dịch vụ thông tin VOInfo (Time_Query_VOInfoService) lớn, nên kéo theo thời gian xử lý toàn bộ tại dịch vụ (Time_VOResDiscoveryService_Processing) lớn. Cuối cùng, thời gian phía client (Time_Client) lớn nhưng trong khoảng này thì thời gian truyền thông nhỏ.

4.3. Nhận xét

Thời gian khám phá tài nguyên là nhỏ (ở phía client trung bình là 250 mili-giây cho mỗi yêu cầu) trong suốt quá trình kiểm tra với 1000 yêu cầu tuần tự. ả hưng sau một số yêu cầu (480 yêu cầu) thì hệ thống chậm ở tầng vận chuyển của mạng kéo theo thời gian khám phá tài nguyên dài hơn bình thường.

5. Kết luận và hướng phát triển

Tóm lại, trong xu thế phát triển tất yếu của công nghệ thì tính toán lưới là một xu hướng mới và cần các kỹ thuật khác hỗ trợ như khám phá tài nguyên lưới. Cho dù trước đây đã có các nghiên cứu về vấn đề khám phá tài nguyên trên lưới [2], [3], [13], [19], nhưng thông qua bài báo này chúng tôi trình bày một kỹ thuật đóng góp thêm vào quan điểm tìm kiếm tài nguyên lưới theo định hướng của người dùng (user-driven resource discovery). Cụ thể, bài báo trình bày kỹ thuật xử lý các yêu cầu mà người dùng xác định các ràng buộc và khám phá tài nguyên lưới trên từng công việc thực thi: (i) có các ràng buộc trên từng máy về bộ xử lý, bộ nhớ, dung lượng trống của đĩa cứng, môi trường thực thi, hệ điều hành…; (ii) hoặc ràng buộc tổng thể như: tổng số bộ xử lý cần cho công việc, số lượng tối đa số tài nguyên trong kết quả trả về.... Bài báo cũng đã trình bày giải thuật set-matching và xếp hạng trong việc khám phá tài nguyên theo yêu cầu của người dùng.

Trong tương lai, giải pháp này có thể được mở rộng cho phép: (i) thông tin được lấy từ nhiều dịch vụ thông tin tài nguyên lưới như từ Ganglia; (ii) mở rộng thêm nhiều ràng buộc về mối quan hệ tiến trình tính toán và dữ liệu; (iii) ứng dụng các kỹ thuật web ngữ nghĩa trong việc định nghĩa các thông tin tài nguyên dưới dạng OWL và động cơ suy diễn theo các luật logic mờ.

6. Lời cảm ơn

Bài báo này được hoàn thành dưới sự tài trợ nghiên cứu trong dự án EDAGrid, và đã nhận được sự góp ý quý báo của các đồng nghiệp nhóm nghiên cứu EDAGrid.

TÀI LIỆU THAM KHẢO

1. ả .T. Anh. *EDAGrid: D1.1. Tài liệu đặc tả công việc (Job Description)*. 26/07/2006.
2. Condor – High Throughput Computing. *http://www.cs.wisc.edu/condor/*
3. *Condor-G. http://www.cs.wisc.edu/condor/manual/v6.4/5_3 Condor_G.html*
4. K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, S. Tuecke. *From OGSI to WSRF: Refactoring and Evolution*. 2004.
5. K. Czajkowski, I. Foster, C. Kesselman, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe. *The WS-Resource Framework*. 03/05/2004.
6. K. Czajkowski, I. Foster, C. Kesselman. *Resource Co-Allocation in Computational Grid*. 2001.
7. ả .C. Đạt và Đ.T. ả ghĩa. *EDAGrid: D4.1. VO Information Service*. 06/09/2006.
8. I. Foster (2005), *A Globus Primer*. (2005).
9. I. Foster và C. Kesselman (2004). *The GRID 2 – Blueprint for a New Computing Infrastructure*. Tái bản lần 2. ả XB Morgan Kaufmann. 2004.
10. I. Foster, C. Kesselman và S. Tueke. *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*. Tạp chí Int. Journal of Supercomputer Applications. pp 200-222. 2001.
11. I. Foster, C. Kesselman, C. Lee, B. Lindell, K. ả ahrstedt và A. Roy. *A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation*. 2002.
12. I. Foster, C. Kesselman, J. ả ick và S. Tueke (2002). *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, Open Grid Service Infrastructure WG*. Global Grid Forum. 2002.
13. Globus Toolkit Information Service: Monitoring & Discovery System (MDS). http://www.globus.org/toolkit/mds/
14. Globus Toolkit. *Web site: http://www.globus.org/*
15. ả .Q. Hùng. *EDAGrid: D2.1. Resource Discovery Interface*. 30/08/2006.
16. IBM Redbook (2005), *Introduction to Grid Computing. 2005*. IBM Corp.
17. ả .T. Son và ả . Q. Hung (2005). *A Practical Grid Service-Oriented Architecture*. Proc. of the International School on Computational Science and Engineering: Theory and Applications, Ho Chi Minh city, Vietnam (March 2005), pp. 115 - 122.
18. B. Sotomayor (2005). *The Globus Toolkit 4's Programmer Tutorial*. http://gdp.globus.org/gt4-tutorial/multiplehtml/index.html
19. H. Tangmunarunkit, S. Decker, C. Kesselman. *Ontology-Based Resource Matching in the Grid – The Grid Meets the Semantic Web*.