Nền tảng chung cho việc phát triển ứng dụng hướng đến môi trường tính toán lưới

Nguyễn Quang Hùng

Khoa Công nghệ Thông tin,Trường đại học Bách Khoa TP.HCM

Tóm tắt

Công nghệ tính toán lưới – Grid Computing, hứa hẹn đem lại một cuộc cách mạng thực sự trong khoa học và cả xã hội. Công nghệ tính toán lưới này sẽ làm thay đổi cách chúng ta suy nghĩ về phần cứng và phần mềm. Tuy nhiên việc phát triển hệ thống và kể cả ứng dụng vẫn còn rất khó khăn và đơn lẻ theo từng trường hợp một, mặc dù hiện nay đã có kiến trúc về môi trường tính toán lưới hay kiến trúc Open Grid Services Architecture (OGSA) để tích hợp các hệ thống phân bố bằng dịch vụ lưới. Do đó, bài báo này đề cập đến việc xây dựng một nền tảng chung cho các ứng dụng cần dùng tài nguyên trên Lưới. Đó là xây dựng ứng dụng e-science và e-business trên kiến trúc lớp hàm ảo. Mỗi tác vụ thuộc lớp hàm ảo có thể được hiện thực bởi tập hợp các dịch vụ lưới trên các nút lưới khác nhau. Để minh chứng cho khả năng hiện thực được, bài báo còn đề cập đến một nghiên cứu đã thành công là xây dựng được môi trường lưới (đặt tên là BK-GRID) dùng bộ Globus Toolkit 3, và đã hiện thực ứng dụng tìm kiếm bằng ngữ nghĩa chạy trên môi trường lưới BK-GRID. Hệ thống lưới BK-GRID chúng tôi xây dựng được bao gồm 8 nút lưới, với mỗi nút lưới là một máy tính PC Pentium IV có bộ nhớ RAM là 256 Mbytes.

Abstract

Grid Computing promises a revolution both science and social. We can be changed thinking about software and hardware by the Grid computing. However, developing a computing system and grid-enabled application based on case by case. Although, there are some proposed standards such as Grid Architecture, OGSA for integrating distributed systems by grid services. This paper proposes a building common virtual Application Programming Interface (API) layer that can apply for e-science or e-business applications. Each virtual operation can be implemented by group of grid services that run on grid nodes. To prove that the virtual API layer can be implemented, this paper also describes about a grid environment called BK-GRID that is built on Globus Toolkit 3. The BK-GRID has eight grid-nodes. Each grid node is a PC Pentium IV that has RAM memory is 256 Mbytes.

1. GIỚI THIỆU

Công nghệ tính toán lưới (Grid Computing) tập trung vào giải quyết các thách thức trong vấn đề chia xẻ tài nguyên khác nhau, phân bố rộng lớn. Công nghệ này lấp vào khoảng trống mà những công nghệ như Internet, CORBA, Enterprise Java Bean hay peer-to-peer không giải quyết triệt để. Hơn một thập kỷ qua, các công nghệ Lưới bao gồm: (1) các giải pháp bảo mật [1, 2, 5] (khóa chung/khóa riêng, chữ ký, đăng nhập một lần…); (2) các dịch vụ và các giao thức quản lý tài nguyên [1, 7, 13]; (3) các dịch vụ và giao thức truy vấn thông tin [1, 6, 14]; và (4) các dịch vụ quản lý dữ liệu [1, 7, 13]. Tất cả những công nghệ này sẽ phù hợp với kiến trúc sẽ mô tả trong phần tiếp.

Kiến trúc Lưới [1] và một framework chung thống nhất cho Lưới là OGSA [8] được công bố. Bên cạnh đó, Globus Toolkit [24] đã và sẽ trở thành một môi trường hiện thực de-facto cho Lưới (trước khi có OGSA). Các phần tử chính trong Globus Toolkit được mang vào chuẩn OGSA và Open Grid Service Infrastructure (OGSI) [12]. Đặc biệt hơn Globus Toolkit 3 đã hiện thực cho OGSA/OGSI phiên bản 1.0 và được chấp nhận rộng rãi trong cộng đồng khoa học và công nghiệp. Bài báo này đề cập đến việc xây dựng một môi trường Lưới thử nghiệm gọi là BK-GRID dựa vào Globus Toolkit 3. BK-GRID bao gồm nhiều nút lưới. Mỗi nút lưới là môi trường hosting và thực thi của các thực thể dịch vụ lưới.

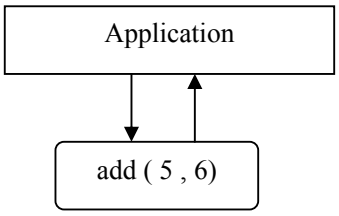
Trong bài báo [9] có đề cập đến một kiến trúc hướng dịch vụ lưới cho ứng dụng tìm kiếm thông tin theo ngữ nghĩa. Theo kiến trúc hướng dịch vụ lưới này, ứng dụng được chia làm nhiều loại dịch vụ lưới với

vai trò khác nhau được triển khai trên BK-GRID. Tất cả tạo nên khái niệm môi trường máy chủ ảo (Virtual Hosting Environment - VHE). Để mở rộng ý tưởng này, trong bài báo này chúng tôi đề cập đến việc tạo ra một tầng hàm ảo khi nhìn từ phía client. Ý nghĩa của tầng hàm ảo này sẽ giúp trong suốt quá trình hiện thực và thực thi một tác vụ được gọi từ phía client. Tác vụ gọi là ảo vì khi client gọi tác vụ này thì nó có thể (không yêu cầu) thực thi trên máy của client, mà lời gọi này sẽ là quá trình gọi tác vụ đến tầng dịch vụ lưới. Tầng dịch vụ lưới sẽ được hiện thực sao cho ý nghĩa của lời gọi tác vụ sẽ được trả về phía client.

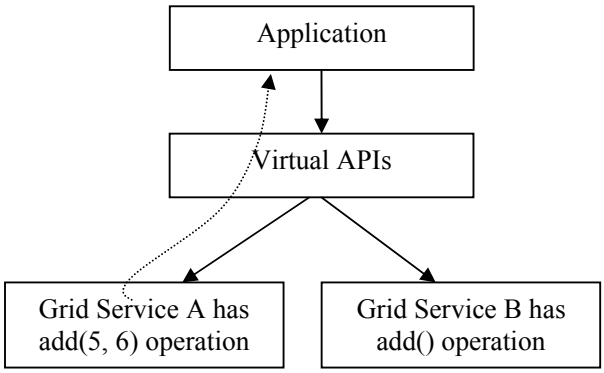
Các phần sau của bài báo sẽ được bố cục như sau: phần 2 trình bày quá trình gọi tác vụ từ lớp hàm ảo trong sự so sánh với quá trình gọi hàm truyền thống; phần 3 trình bày về việc hiện thực lớp hàm ảo bằng kiến trúc hướng dịch vụ lưới; phần 4 trình bày về các số liệu thử nghiệm; phần cuối cùng đưa ra kết luận và hướng phát triển.

2. SO SÁNH QUÁ TRÌNH GỌI HÀM TRUYỀN THỐNG VÀ GỌI TÁC VỤ TỪ LỚP HÀM ẢO

Quá trình gọi hàm truyền thống và gọi tác vụ từ lớp hàm ảo được minh họa bằng hình 2-1 và hình 2-2 bên dưới.



Hình 2-1: Lời gọi hàm truyền thống



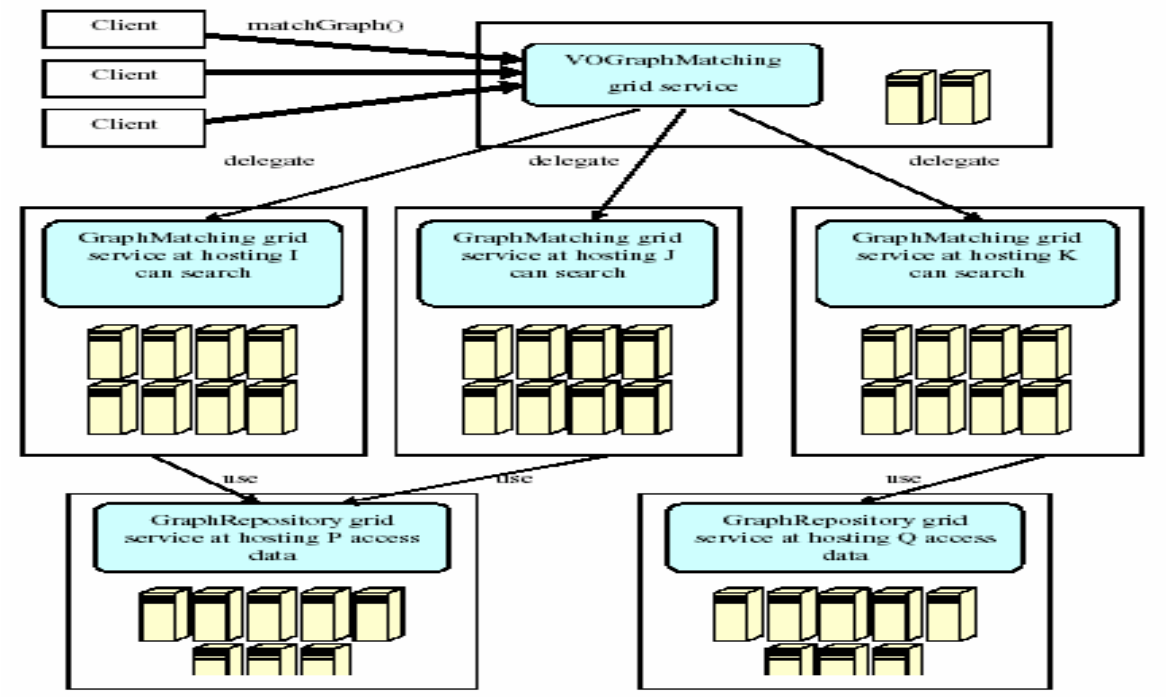
Hình 2-2: Quá trình gọi một tác vụ từ tầng hàm ảo.

Trong quá trình gọi hàm truyền thống (hình 2-1), lời gọi hàm add(5,6) sẽ được thực thi ngay trên máy chạy ứng dụng. Còn trong trong quá trình gọi một tác vụ từ lớp hàm ảo (hình 2-2), một thực thể dịch vụ lưới A hoặc B được tạo ra (tùy vào chiến lược lựa chọn) và sau đó gọi hàm add(5,6).

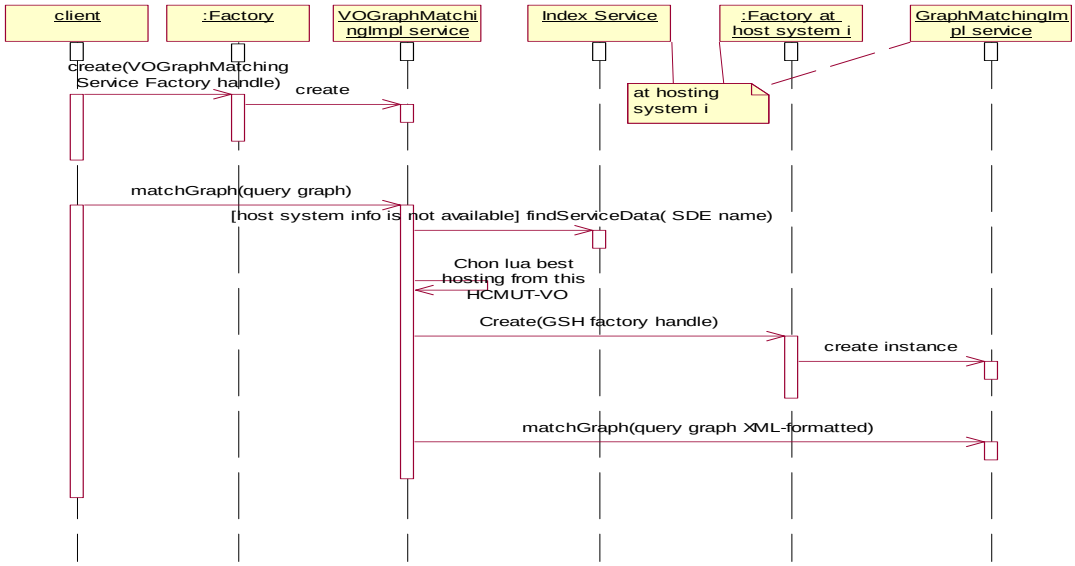
Lớp hàm ảo (Virtual APIs) này sẽ là tập hợp các tác vụ (operation) mà các dịch vụ lưới hỗ trợ. Ví dụ, trong ứng dụng tìm kiếm theo ngữ nghĩa đã được hiện thực trong [9] thì các dịch vụ lưới hỗ trợ một tác vụ là matchGraph() – với thông số đầu vào là chuỗi XML mô tả một đồ thị biểu diễn câu truy vấn tìm kiếm và đầu ra là chuỗi XML mô tả kết quả.

3. HIỆN THỰC LỚP HÀM ẢO BẰNG KIẾN TRÚC HƯỚNG DỊCH VỤ LƯỚI

Trong bài báo [9], chúng tôi đã đề nghị một kiến trúc hướng dịch vụ lưới cho ứng dụng tìm kiếm theo ngữ nghĩa như hình 3-1. Quá trình hiện thực cách thức gọi tác vụ so trùng matchGraph() được hiện thực theo lược đồ trình tự như hình 3-2.



Hình 3-1: Kiến trúc hướng dịch vụ cho ứng dụng tìm kiếm theo ngữ nghĩa.



Hình 3-2: Lược đồ trình tự biểu diễn quá trình gọi tác vụ so trùng matchGraph()

*** CÁC CHIẾN LƯỢC CÂN BẰNG TẢI TÍNH TOÁN CHO CÁC NÚT LƯỚI**

Trong quá trình tìm kiếm một dịch vụ hiện thực tác vụ so trùng matchGraph(), thực thể của dịch vụ lưới VOGraphMatching (đóng vai trò VHE Master) phải quan tâm đến sự cân bằng tải của các nút lưới có triển khai các dịch vụ tính toán GraphMatching (đóng vai trò VHE Worker). Do vậy bài báo [9] cũng đề nghị một số giải thuật cân bằng tải như là: phân phối xoay vòng các nút lưới theo round robin, lựa chọn nút lưới tốt nhất. Cả hai chiến lược này đã được hiện thực và đo đạc.

Tuy nhiên để thực hiện được quá trình cân bằng tải tốt nhiều vấn đề đặt ra: (1) phải biết thông tin hệ thống của các nút lưới (như là tải rảnh của CPU, dung lượng bộ nhớ và số Mbytes còn trống của RAM, băng thông mạng…); (2) ước lượng thời gian thực thi của từng yêu cầu tìm kiếm; (3) số lượng các yêu

cầu tìm kiếm nằm trong hàng đợi của một nút lưới. Trong ba yêu cầu này thì chúng tôi chỉ giải quyết được hai điều là: phát triển được một phương thức truy vấn thông tin hệ thống trên từng nút lưới và thống kê số lượng các yêu cầu tìm kiếm đã giao cho một nút lưới. Phương pháp truy vấn thông tin tài nguyên của từng nút lưới được hiện thực dưới dạng một Information Provider và cắm vào dịch vụ lưới Index của Globus Toolkit 3.

4. THỬ NGHIỆM

Trong phần thử nghiệm, chúng tôi đã hiện thực lớp hàm ảo bằng sự phối hợp của các dịch vụ lưới. Trong quá trình phối hợp chúng tôi sử dụng hai chiến lược cân bằng tải tính toán trên các nút lưới theo thông tin về tải rảnh của CPU, dung lượng còn trống của bộ nhớ RAM và số lượng yêu cầu so trùng trong hàng chờ thực thi tại nút lưới. Tiêu chí đánh giá trên từng chiến lược cân bằng tải là đo thời gian cho cả quá trình từ lúc client gọi tác vụ so trùng matchGraph() đến khi kết quả được trả về cho client. Các biểu đồ bên dưới so sánh hai chiến lược này qua cả hai trường hợp sau:

4.1. Trường hợp 1

Hệ thống BK-GRID có cấu hình gồm:

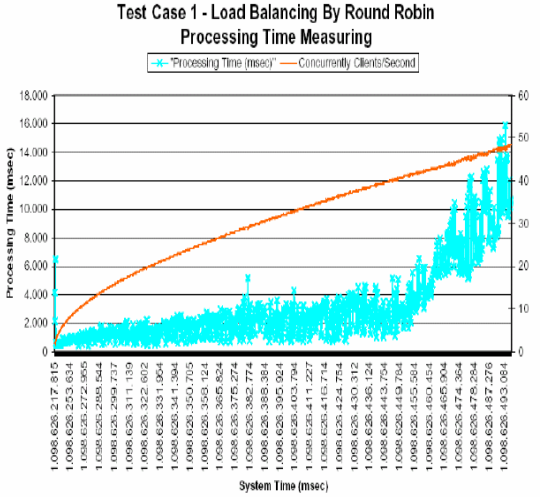
Phần cứng: bao gồm 8 nút lưới là một máy PC có 1 CPU, 256 Mbytes RAM.

Phần mềm: các nút lưới đều cài hệ điều hành là RedHat Enterprise Server version 3, Globus Toolkit version 3.2. Trong đó có 1 máy sẽ host dịch vụ lưới VOGraphMatching, 4 máy sẽ host dịch vụ GraphMatching, 1 máy có Xindice XML database server, 1 máy làm NIS server, 1 máy giả lập các client truy xuất tác vụ so trùng matchGraph().

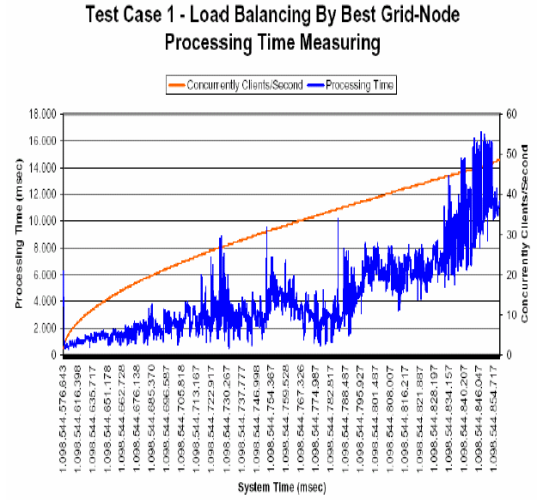
Hoạt động: ban đầu tất cả các nút lưới đều không phục vụ ứng dụng cả và trong suốt quá trình thử nghiệm chỉ chạy ứng dụng so trùng này (môi trường lý tưởng). Minh họa ở hình 4-1a và hình 4-1b.

4.2. Trường hợp 2

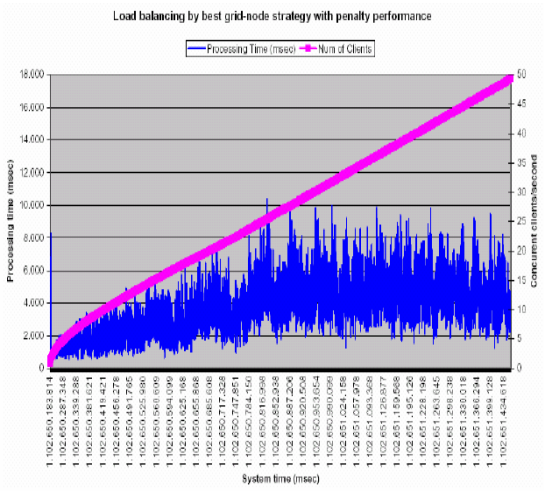
Hệ thống BK-GRID có cấu hình phần cứng và phần mềm giống như trường hợp 1. Tuy nhiên, về hoạt động ban đầu các nút đều rảnh và trong quá trình thử nghiệm sẽ có nhiều người khác chạy các chương trình khác nhau trên các máy tham gia vào hệ thống (môi trường thực). Minh họa ở hình 4-2a và 4-2b.



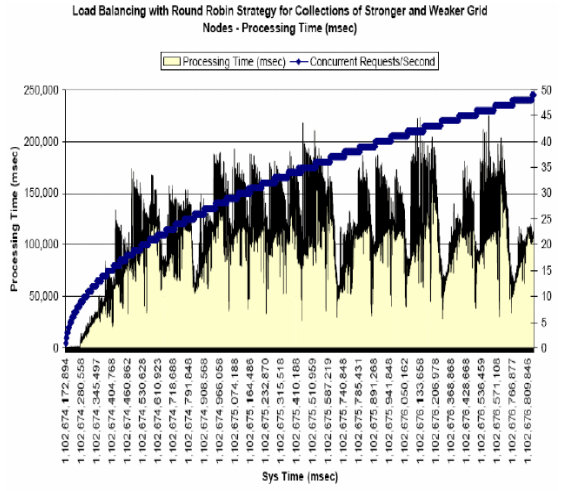
Hình 4-1a: Trường hợp 1: Thời gian xử lý từng yêu cầu so trùng của hệ thống trong chiến lược phân phối xoay vòng (round-robin).



Hình 4-1b: Trường hợp 1. Thời gian xử lý từng yêu cầu so trùng của hệ thống trong chiến lược tìm nút lưới tốt nhất.



Hình 4-2a: Trường hợp 2. Thời gian xử lý từng yêu cầu so trùng của hệ thống trong chiến lược tìm nút lưới tốt nhất.



Hình 4-2b: Trường hợp 2. Chiến lược phân phối xoay vòng.

Qua số liệu thống kê cho thấy trong trường hợp 1 thì chiến lược phân phối xoay vòng (round-robin) là tốt hơn chiến lược tìm nút lưới tốt nhất là vì không mất thời gian tìm thông tin về nút lưới và không có tình trạng một nút lưới bị giao nhiều công việc so trùng (trong thời gian rất ngắn). Còn trong môi trường thực tế nhiều người dùng thì chiến lược tìm nút lưới là tốt hơn cả và chiến lược xoay vòng có thể làm hệ thống treo lâu vì khi đó nút lưới đã có tải nặng có thể lại bị giao cùng số công việc như nút lưới tải nhẹ.

5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kiến trúc dùng lớp hàm ảo cung cấp các tác vụ cần thiết cho ứng dụng phân bố trên môi trường Lưới là rất uyển chuyển và là xu hướng phát triển mới. Bài báo này cho thấy lớp hàm ảo đem lại các lợi ích sau:

Thứ nhất: ứng dụng không cần phụ thuộc vào ngôn ngữ hiện thực tác vụ trong quá trình ứng dụng thực thi (đây cũng là xu hướng kết dính động – dynamic binding). Miễn là có thể phát triển dịch vụ lưới bằng nhiều ngôn ngữ lập trình khác nhau và môi trường thực thi của dịch vụ lưới hỗ trợ điều này. Hiện tại Globus Toolkit 4 trong tương lai sẽ cho phép phát triển dịch vụ lưới bằng ngôn ngữ Java, C++; và cũng đã có nhiều phiên bản hiện thực chuẩn OGSA/OGSI cho phép hiện thực dịch vụ lưới bằng ngôn ngữ C#.

Thứ hai: mang lại sự trong suốt về yếu tố chất và lượng của các tài nguyên trên môi trường Lưới. Số lượng các tài nguyên có thể tăng thêm động, và mỗi nút lưới có thể được thay đổi (về cấu hình mạng, từ một máy đơn đến hệ thống Cluster) trong suốt với phía sử dụng tài nguyên.

Tóm lại sự thành công trong tương lai sẽ phụ thuộc rất nhiều vào sự mở rộng của hệ thống BK-GRID, số lượng các nhà cung cấp tác vụ tương thích chuẩn OGSA/OGSI (có thể mở rộng nhà cung cấp dịch vụ qua mạng Internet). Trong tương lai chắc chắn sẽ có nhiều nhà cung cấp quan tâm đến việc cung cấp các tác vụ cho ứng dụng khoa học và doanh nghiệp.

LỜI CÁM ƠN

Xin chân thành cám ơn TS Nguyễn Thanh Sơn (sonsys@hcmut.edu.vn) đã hướng dẫn hoàn thành luận văn Thạc sỹ. Xin chân thành cám ơn tập thể cộng đồng mã nguồn mở Globus Toolkit (www.globus.org).

TÀI LIỆU THAM KHẢO

1. Ian Foster, Carl Kesselman, Steven Tuecke. *The Anatomy of the Grid Enabling Scalable Virtual Organizations,* Initl J. SuperComputing Applications, 2001.

2. I. Foster and C.Kesselman (Eds). *The Grid 2 – Blueprint for New Computing Infrastructure*, trang 5-10, 2004.
3. Ann Chervenak, Jan Foster, Carl Kesselman, Charles Salisbury, Steven Tuecke, *The Data Grid – Towards an architecture for distributed management and analysis of large scientific Dataset*.
4. Haiping Zhu, Jiwei Zhong, Jianming Li, Yong Yu. *An Approach for Semantic Search by Matching RFD graphs*, American Association for AI Publisher, 2002.
5. Foster, I. and Kesselman, C. , Tsudik, G. and Tuecke, S. *A Security Architecture for Computational Grids*, trong ACM Conference on Computers and Security, trang 83-91, 1998.
6. Czajkowski, Fitzgerald, Foster, Kesselman. *Grid Information Services for Distributed Resource Sharing*, 2001.
7. Czajkowski, Ian Foster, Karonis, Kesselman, Martin, Smith, Tuecke. *A Resource Management Architecture for Metacomputing Systems*. Trong The 4th Workshop on Job Scheduling Strategies for Parallel Processing, trang 62-68, 1998.
8. Ian Foster, Carl Kesselman , Jeffrey M. Nick, Steven Tuecke. *The Physiology of the Grid – An Open Grid Services Architecture for Distributed Systems Integration*. 2002.
9. Nguyen Thanh Son và Nguyen Quang Hung. *A Practical Grid Service-Oriented Architecture*. Hội nghị COSCI 2005, trường Đại học Bách Khoa TP.HCM. 2005.
10. IBM Redbook – Globus Toolkit 3 Quick Start, REDP3697, August 2003.
11. IBM Red book – Fundamental of Grid computing, Viktors Berstis, 2002.
12. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt. *Open Grid Service Infrastructure*. Version 1. 27/06/2003.
13. Karl Czajkowski, Ian Foster, Carl Kesselman. *Resource Co-Allocation in Computational Grid*. 2001.
14. Ian Foster, Carl Kesselman, Craig Lee, Bob Lindell, Klara Nahrstedt, Alain Roy. *A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation*. 2002.
15. Karl Czajkowski, Ian Foster, Carl Kesselman, Steve Graham, Igor Sedukhin, David Snelling, Steve Tuecke, William Vambenepe. *The WS-Resource Framework*. 03/05/2004.
16. Karl Czajkowski, Don Ferguson, Ian Foster, Jeff Frey, Steve Graham, Tom Maguire, David Snelling, Steve Tuecke. *From OGSI to WSRF: Refactoring and Evolution.* 2004.
17. Jean Pierre Goux, Sanjeev Kulkarni, Jeff Linderoth, Michael Yoder. *An Enabling Framework for Master-Worker Applications on the Computational Grid*. 2001.
18. Gregor von Laszewski, Ian Foster, Jarek Gawor, Peter Lane, Nell Rehn, Mike Russell. *Designing Grid-Based Solving Environment and Portals*. 2000.
19. Gregor Von Laszewski, Ian Foster. *Usage of LDAP in Globus*. 2000.
20. Borja Sotomayor. *Globus Toolkit 3 Programmer's Tutorial*. 2004.
21. Globus group. *Java programmer's Guide Core Framework*. 03/09/2003.
22. Li, J., Zhang, L., and Yu, Learning to Generate Semantic Annotation for Domain Specific Sentences. 2001.
23. Sowa, J.F. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, 1984.
24. Globus Toolkit 3.0 tại http://www.globus.org/
25. IBM developers website: http://www.ibm.com
26. W3C RDF web site: http://www.w3c.org/RDF