ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

ꕥ <> ꕥ

Đề tài:

GRID RESOURCE MANAGEMENT

GV: **TS. Phạm Trần Vũ**

SV: **Nguyễn Phan Thiện Bách** **09070421**

Trần Minh Hùng **10070481**

Nguyễn Mạnh Tuấn **10070504**

TP. HỒ CHÍ MINH

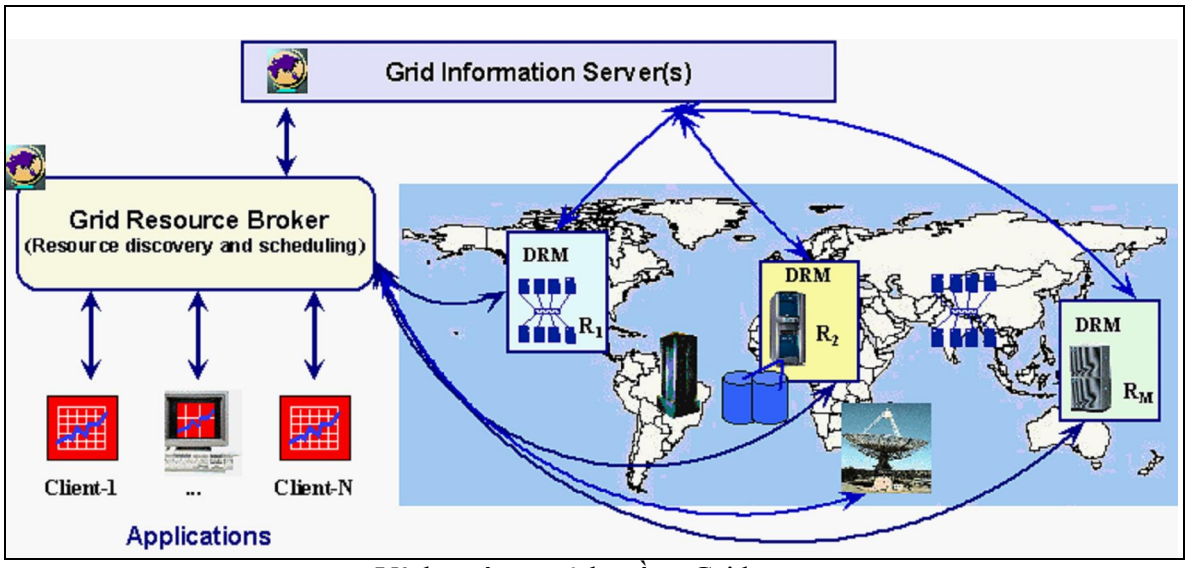
THÁNG 06/2011

MỤC LỤC

1 Grid Resource Management

Sự phát triển ngày càng phổ biến của Internet, cùng với năng lực tính toán của máy tính ngày càng mạnh và mạng tốc độ cao cũng như các thiết bị có chi phí ngày càng thấp đang thay đổi cách tính toán và sử dụng máy tính. Các tài nguyên trong Grid, thường được phân bố theo vị trí địa lý khác nhau, cần được liên kết với nhau để giải quyết các bài toán quy mô lớn (large-scale problems) và cần phải có một chiến lược để quản lí chúng một cách hiệu quả.

Quản lý tài nguyên trong Grid là khả năng tìm kiếm, đàm phán và cấp phát việc sử dụng các tài nguyên được chia sẻ trong môi trường mạng. Tài nguyên ở đây không chỉ được hiểu theo nghĩa hẹp như một thực thể vật lý, chẳng hạn: máy tính, mạng hoặc hệ thống lưu trữ,… mà là tất cả các thành phần có thể được chia sẻ và khai thác trong môi trường mạng, bao gồm các thực thể nói trên và các thành phần khác như phần mềm, dịch vụ, con người, tri thức,…



Ví dụ một cơ sở hạ tầng Grid

Vậy quản lý tài nguyên trong Grid có những điểm gì khác với quản lý tài nguyên trong các hệ thống truyền thống? Trong các hệ thống truyền thống, các tài nguyên được phân bố cục bộ, ta có toàn quyền kiểm soát chúng, vì vậy các cơ chế và chính sách để sử dụng hiệu quả các tài nguyên là do ta quyết định. Tuy nhiên tài nguyên trong Grid được phân tán và được quản lý bởi các tổ chức khác nhau, có các chính sách quản lý khác nhau, vì vậy có nhiều vấn đề phức tạp phát sinh trong việc liên kết các tài nguyên này. Dưới đây là một số tình huống quản lý tài nguyên đa dạng có thể xảy ra:

- **Task submission**: tài nguyên nhận thực hiện một tác vụ cụ thể, ví dụ, thực thi một chương trình, di chuyển một tập tin, hoặc thực hiện một truy vấn cơ sở dữ liệu . Đây là loại thỏa thuận quản lý tài nguyên đơn giản nhất, trong đó nhà cung cấp chỉ cam kết thực hiện tác vụ mà không nhất thiết phải cam kết khi nào tác vụ sẽ bắt đầu và kết thúc, bao nhiêu tác vụ mà tài nguyên có thể chấp nhận tại thời điểm đó, bao nhiêu tác vụ có thể thêm vào trong tương lai, ...

- **Workload management**: thứ tự thực hiện các tác vụ trong task submission có thể được mở rộng bằng cách thêm vào dịch vụ gán tác vụ với mức độ khả năng (level of capacity), chẳng hạn như bộ vi xử lý trên một máy tính, các luồn xử lý hay bộ nhớ trong một máy chủ, băng thông mạng, hay không gian đĩa trên hệ thống lưu trữ. Workload management cho phép khách hàng không chỉ kiểm soát những công việc sẽ được thực hiện mà còn theo dõi việc chúng được thực hiện như thế nào. Mức độ khả năng có thể xác định thông qua thời gian quay vòng tác vụ tối đa, thời gian quay vòng trung bình, thông lượng tác vụ …
- **On-demand access**: khả năng tài nguyên luôn có sẵn tại một thời điểm xác định, hoặc trong một khoảng thời gian xác định. Loại quản lý tài nguyên này đặc biệt quan trọng cho các ứng dụng trực tuyến, chẳng hạn như teleoperation trong NEESgrid.
- **Coscheduling**: tập hợp các tài nguyên có sẵn đồng thời bằng cách phối hợp các thỏa thuận theo yêu cầu giữa các tài nguyên cần thiết. Sử dụng loại quản lý này cho các dịch vụ truyền dữ liệu (các hệ thống lưu trữ đầu cuối phải phối hợp cùng với băng thông mạng), hoặc phân phối các tính toán song song (nhiều tài nguyên tính toán có sẵn ở cùng thời điểm).
- **Resource brokering**: một dịch vụ môi giới hoạt động như thành phần trung gian đến tập các tài nguyên và ánh xạ các tác vụ vào các tài nguyên thích hợp dựa trên các chính sách môi giới cụ thể. Một trong những chính sách này là tối đa hoá số tác vụ được thực thi.

Tóm lại, quản lý tài nguyên Grid không chỉ đảm bảo việc thực thi tác vụ mà còn đảm bảo chất lượng của việc thực thi này (quality of service). Hệ thống quản lí tài nguyên Grid không thể đảm bảo chất lượng dịch vụ mà không có sự hợp tác giữa các tài nguyên đang được quản lý. Nguyên nhân là các tài nguyên thường không chỉ thuộc về một tổ chức ảo mà được chia sẻ giữa nhiều tổ chức ảo với nhau hoặc là giữa người dùng trong Grid và người dùng không thuộc Grid.

Một dịch vụ quan trọng mà hệ thống quản lí tài nguyên Grid cần cung cấp cho người dùng là dịch vụ đặt chỗ tài nguyên: khả năng yêu cầu tài nguyên trước khi đưa tác vụ vào thực hiện.

Để sử dụng tài nguyên một cách hiệu quả, hệ thống quản lí tài nguyên cần quan tâm đến các vấn đề sau: lợi nhuận, tiết kiệm năng lượng, chi phí quản lí, tính công bằng trong việc cấp phát tài nguyên. Nhằm giải quyết các vấn đề này, hệ thống cần phải thực hiện các công việc sau: lập kế hoạch sử dụng tài nguyên, triển khai, giám sát và đánh giá hiệu quả sử dụng tài nguyên.

Hiện nay có nhiều hướng tiếp cận khác nhau để phát triển hệ thống quản lí tài nguyên Grid. Việc lựa chọn mô hình kiến trúc quản lý tài nguyên phù hợp đóng một vai trò lớn đến hiệu quả của việc sử dụng các tài nguyên. Dưới đây sẽ giới thiệu ba mô hình kiến trúc khác nhau cho việc quản lý tài nguyên lưới: Hierarchical Model, Abstract Owner Model và Computational Market/Economy Model

Mô hình	Đặc trưng	Các hệ thống
Hierarchical	Mô hình được áp dụng phổ biến trong các hệ thống hiện tại. Quản lý tài nguyên theo cơ chế phân cấp và lập lịch sử dụng các tài nguyên	Globus, Legion, Ninf, NetSolve
Abstract Owner	Áp dụng mô hình yêu cầu và chuyển giao (*order and delivery model*) trong việc chia sẻ tài nguyên hướng đến hiệu quả về chi phí, bỏ qua cơ sở hạ tầng hiện có để tập trung vào mục tiêu dài hạn.	Expected to emerge
Economy / Market	Áp dụng mô hình kinh tế trong việc phát hiện và lập kế hoạch sử dụng resource (*resource discovery and scheduling*). Sử dụng kết hợp cả hai mô hình *hierarchical và abstract owner models*.	Nimrod/G, JaWS, Myriposa, JavaMarket

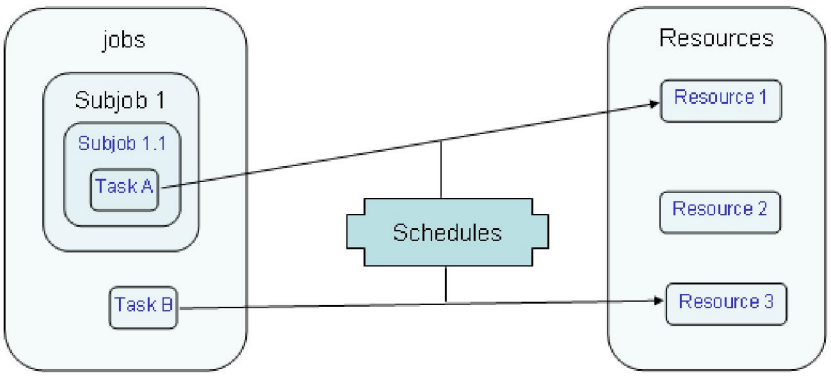
Ba mô hình cho Kiến trúc quản lý tài nguyên Grid

2 Hierarchical Model

Mô hình này được đưa ra trong cuộc họp lần 2 của Grid Forum vào tháng 11 năm 1999. Các thành phần chính của kiến trúc này được chia thành hai loại: các thành phần thụ động (*passive components*) và các thành phần tích cực (*active component*).

2.1 Các thành phần thụ động: bao gồm

- ***Resources*** (Tài nguyên) là những thứ có thể được sử dụng trong một khoảng thời gian, và có thể hoặc không thể được gia hạn sử dụng, chủ sở hữu tài nguyên có thể yêu cầu người sử dụng trả phí khi sử dụng tài nguyên. Tài nguyên có thể được chia sẻ hoặc không, có thể được đặt tên rõ ràng, hoặc được mô tả dạng thông số.
- ***Tasks*** (Tác vụ) là đối tượng trực tiếp sử dụng tài nguyên.
- ***Jobs*** (Công việc) là các thực thể có thứ bậc, có thể có cấu trúc đệ quy. Một công việc có thể bao gồm nhiều công việc con hoặc các tác vụ, các công việc con này có thể gồm nhiều công việc con khác. Công việc đơn giản nhất là công việc chỉ bao gồm một tác vụ.
- ***Schedules*** là những ánh xạ của các ***task*** vào các ***resource*** theo thời gian.



Các thành phần thụ động của mô hình Hierarchical

2.2 Các thành phần tích cực: bao gồm

- ***Users*** (người sử dụng) gửi *jobs* đến *Resource Management System*.
- ***Job Control Agents*** chịu trách nhiệm quản lý một *job* trong hệ thống, có thể hoạt động như là một *proxy* cho *user* và như là một điểm kiểm soát liên tục cho *job*. Nhiệm vụ của các *job control agent* là phối hợp giữa các thành phần khác nhau trong *resource management system*, ví dụ như phối hợp giữa các *monitors* và *schedulers*.
- ***Admission Control Agents*** xác định xem hệ thống có thể phục vụ thêm *jobs*, và ra quyết định từ chối hoặc trì hoãn *jobs* khi hệ thống được bão hòa.
- ***Information Services*** hoạt động như cơ sở dữ liệu để cung cấp các thông tin về các thành phần cần quan tâm của hệ thống quản lý tài nguyên như *resources*, *jobs*, *schedulers*, *agents*, ...
- ***Schedulers*** tính toán một hoặc nhiều *schedules* cho các danh sách *jobs* được đưa vào hệ thống, lệ thuộc vào các ràng buộc được quy định tại thời gian chạy. Chúng còn được gọi là *metaschedulers* hoặc *super schedulers*, không thể trực tiếp cấp phát tài nguyên.
- ***Deployment Agents*** thực thi *schedules* bằng cách đàm phán với các *domain control agents* để có được *resource* và bắt đầu thực hiện *task*.
- ***Domain Control Agents*** có thể cấp phép để sử dụng *resource*, có thể gọi là *local resource manager*. *Domain control agents* có thể cung cấp thông tin trạng thái của resource thông qua việc đưa đến *Information Service* hoặc trả lời truy vấn trực tiếp.Ngoài ra nó còn hỗ trợ thêm chức năng *reservation.* Một số các domain control agent thực tế có thể kể ra như Maui Scheduler, Globus GRAM và Legion Host Object.
- ***Monitors*** theo dõi tiến trình thực hiện của *job*. *Monitors* lấy thông tin trạng thái của job từ các *task* thuộc về job đó và từ *Domain Control Agents* nơi các *task* đang chạy. Căn cứ vào các thông tin này, *Monitor* có thể thực hiện *outcalls* đến *Job Control Agent* và *Schedulers* để thực hiện ánh xạ lại *job.*

Việc phân biệt các thành phần trên trong hệ thống chỉ mang tính tương đối, ví dụ trong thực tế schedulers có thể đồng thời thực hiện các công việc của deployment agents hoặc monitors.

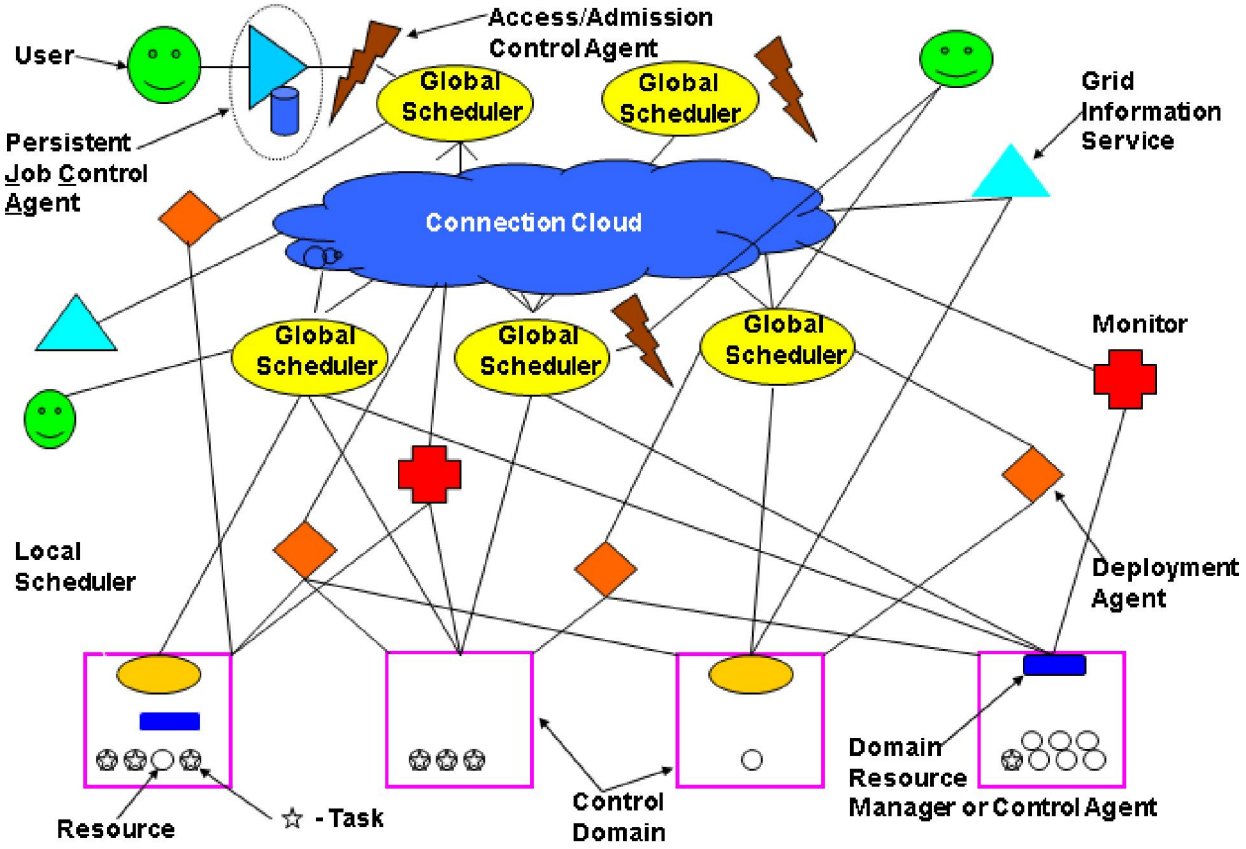
2.3 Tương tác giữa các thành phần:

Dưới đây ta xem xét một ví dụ về sự tương tác giữa các thành phần nêu trên:

User gửi *job* đến *job control agent*, và *job control agent* kế tiếp gọi *admission agent. Admission agent* xem xét các nhu cầu *resource* của *job* đó (lấy thông tin các resource từ *grid information system*) và xác định hệ thống có an toàn khi thêm *job* vào không gian làm việc hiện tại của hệ thống hay không. Nếu thỏa mãn, *Admission agent* chuyển *job* đến một *Scheduler*, tại đây sẽ thực hiện *resource discovery* bằng cách sử dụng *grid information system*

và sau đó liên lạc với các *domain control agents* để xác định trạng thái hiện tại và tính sẵn sàng của các *resources*.

Scheduler tính toán một tập hợp các ánh xạ các *task* vào các *resource* và gửi những ánh xạ đó cho *deployment agent*. *Deployment agent* thương lượng với *domain control agents* cho các *resource* ghi trong *schedule*, và đặt trước việc sử dụng các *resources*. Thông tin này được chuyển đến các *job control agents.* Vào thời điểm thích hợp, các *job control agents* làm việc với một *deployment agent* khác và các *deployment agent* phối hợp với các *domain control agents* thích hợp để bắt đầu thực hiện *task*. *Monitor* theo dõi tiến trình thực hiện *job* và có thể quyết định reschedule nếu hiệu suất thấp hơn dự kiến.



Mô hình thứ bậc cho quản lý tài nguyên lưới.

Ví dụ trên là một trong những cách các thành phần của hệ thống hợp tác với nhau. Khi triển khai, có thể căn cứ vào tình huống cụ thể mà có thể bỏ bớt một số thành phần hoặc kết hợp nhiều thành phần thành một thành phần chung.

3 Mô hình Abstract Owner

Mô hình Abstract Owner được đưa ra như một mô hình lý tưởng nhằm trừu tượng hóa các tài nguyên trong Grid. Để có thể hiểu ý tưởng của việc đề xuất ra mô hình AO, ta xem xét ví dụ sử dụng dịch vụ internet ADSL dành cho cá nhân. Ai sẽ là người sở hữu những tài nguyên

như đường truyền, cáp, router, …? Thực tế người dùng không cần phải quan tâm đến việc này. Điều họ cần biết là có một đối tượng nào đó cung cấp dịch vụ, và họ sẽ thỏa thuận với đối tượng đó để sử dụng dịch vụ.

Trở lại với mô hình AO, trong mô hình này, mỗi một tài nguyên trong grid sẽ được đại diện bởi một hoặc nhiều "sở hữu trừu tượng – abstract owners". Tài nguyên này về độ lớn có thể chỉ là một bộ xử lý hay là cả một hệ thống grid. Với những tài nguyên phức hợp, thì sở hữu trừu tượng này chính xác sẽ làm công việc của một môi giới (broker) trong grid để đàm phán tài nguyên với những sở hữu trừu tượng khác; người sử dụng tài nguyên (grid user) sẽ chỉ cần biết một kênh giao tiếp duy nhất là với sở hữu trừu tượng cung cấp tài nguyên mà họ cần mà không cần quan tâm tài nguyên đó được tổ chức như thế nào.

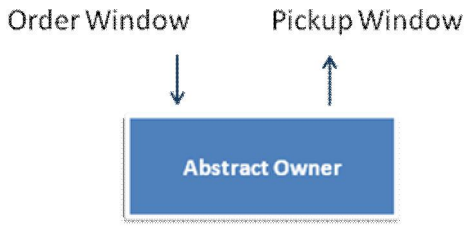
Các phần tiếp theo của mục này sẽ miêu tả chi tiết hơn về sở hữu trừu tượng, tài nguyên, phương thức mà người sử dụng tài nguyên sẽ đàm phán với sở hữu trừu tượng để sử dụng tài nguyên, cách thức người sử dụng sử dụng tài nguyên. Kể từ phần này trở đi, người sử dụng tài nguyên sẽ được gọi là client (có thể là một chương trình máy tính sử dụng tài nguyên hoặc cũng có thể là con người). Sở hữu trừu tượng được gọi là abstract owner (AO) để tránh nhầm lẫn.

3.1 Cấu trúc tổng quát của mô hình AO :

Mô hình của một AO tương tự như cơ cấu của một cửa hàng thức ăn nhanh: có một đầu vào để nhận đặt hàng của khách hàng và một đầu ra để trả kết quả. Để yêu cầu sử dụng một tài nguyên nào đó, client sử dụng Order Window (cửa sổ đặt hàng) để thực hiện việc đàm phán với AO.

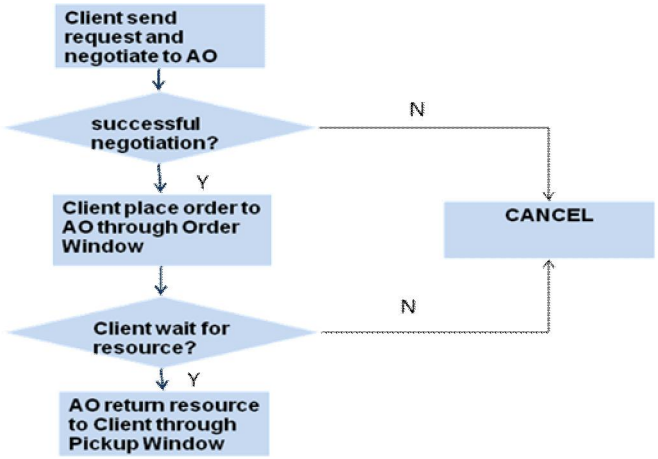
Quá trình đàm phán nhằm xác định khoảng thời gian mà client có thể sử dụng được tài nguyên, hay chi phí để sử dụng tài nguyên đó … Sau khi quá trình đàm phán kết thúc, client có thể thực hiện việc "đặt hàng" tài nguyên hoặc hủy bỏ quá trình đàm phán do điều kiện mà AO đưa ra không thỏa mãn với yêu cầu của client (VD: chi phí cao, không thỏa mãn thời điểm sử dụng tài nguyên).

Nếu việc "đặt hàng" được thực hiện thì sau đó client có thể nhận tài nguyên từ AO thông qua Pickup Window (cửa sổ giao hàng). Việc nhận tài nguyên được thực hiện bằng những giao thức đã được thỏa thuận trước giữa client và AO – chẳng hạn như client có thể đến và nhận tài nguyên vào một thời điểm định trước, hoặc AO phải phát thông điệp thông báo cho client biết khi mà tài nguyên đã sẵn sàng. Các giao thức này sẽ được xác định trong quá trình đàm phán.



External View of AO model

Quá trình đàm phán giữa client và AO được miêu tả qua lưu đồ sau:



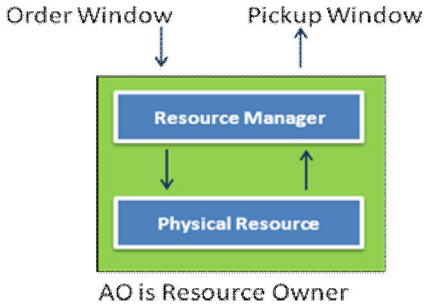
Quá trình đàm phán và nhận tài nguyên

Như vậy, các cửa sổ Order và Pickup là các interface của AO để client có thể access và thực hiện các bước "đặt hàng" cũng như nhận về tài nguyên. Các cửa sổ này phải hỗ trợ những giao thức chuẩn để client có thể trao đổi thông tin. Các luồng thông tin trao đổi giữa client và AO thông thường không đòi hỏi lượng dữ liệu lớn, do đó việc tối ưu hóa hiệu năng truyền nhận không thật sự cần thiết. Ở đây, ta có thể sử dụng những phương thức điều khiển từ xa như CORBA, RPC, RMI, … để hiện thực.

Khái niệm tài nguyên trong mô hình này sẽ được xem như là một đối tượng, với các thuộc tính, các phương thức và giá trị để định danh đối tượng tài nguyên. Thuộc tính của tài nguyên cho phép client có thể tùy biến tài nguyên còn phương thức của tài nguyên được sử dụng để khởi tạo cũng như điều khiển tài nguyên đó. Thông thường, các thuộc tính của tài nguyên sẽ được gán trong quá trình "đặt hàng" và sẽ được truy vấn (chỉ đọc) sau khi tài nguyên đã được chuyển giao.

Nhìn từ bên ngoài, một AO sẽ có cấu trúc chính là 2 interface Order Window và Pickup Window như đã đề cập ở trên, dù AO đó là trực tiếp điều khiển tài nguyên hay là một broker sử dụng những nguồn tài nguyên khác. Tuy nhiên, với góc nhìn từ bên trong, cấu trúc của một AO sẽ có phần khác biệt tùy theo AO đó trực tiếp quản lý tài nguyên hay là broker.

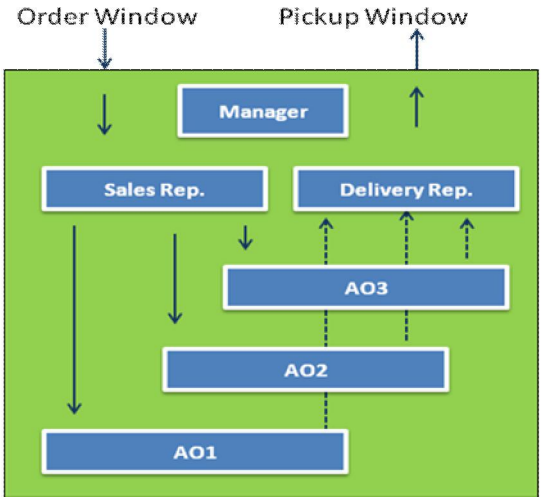
Nếu AO trực tiếp quản lý tài nguyên thì cấu trúc khá đơn giản, nó cần có them một bộ quản lý tài nguyên (Resource Manager) có chức năng đàm phán với client, điều phối tài nguyên và giao tài nguyên cho client sử dụng.



AO trực tiếp sở hữu tài nguyên

Trong trường hợp AO là một broker, nghĩa là AO cần sử dụng tài nguyên của các AO khác để cung cấp thì ngoài thành phần Manager, AO đó cần có thêm 2 thành phần nữa:

- **Sale Representative**: Giao tiếp với các Order Window của các AO khác để đàm phán xin sử dụng tài nguyên
- **Delivery Representative**: Giao tiếp với các Pickup Window của các AO khác để nhận tài nguyên.



AO là một broker

3.2 Tài nguyên trong Grid :

Trong mô hình AO, các nguồn tài nguyên được chia làm 3 loại: Instrument, Channel và Complex.

- **Instrument:** Tài nguyên instrument được định nghĩa là một tài nguyên hiện hữu tại một vị trí hoặc trong một khoảng thời gian mà tài nguyên đó tạo ra, tiêu thụ hoặc chuyển tải dữ liệu, thông tin. Tài nguyên instrument được chia ra làm 3 loại:
  - Compute Instrument: bao gồm các bộ xử lý đơn hoặc đa xử lý với bộ nhớ và các cơ chế lưu trữ tương ứng.
  - Archival Instrument: bao gồm các đối tượng lưu trữ thông tin mà đơn vị nhỏ nhất là file.
  - Personal Instrument: bao gồm các interface mà cần có sự tham gia trực tiếp của con người
- **Channel:** Tài nguyên dạng Channel là các tài nguyên nhằm phục vụ việc truyền tải dữ liệu hoặc thông tin giữa hai hay nhiều instrument giữa những địa điểm khác nhau hay cùng một địa điểm ở những thời điểm khác nhau. Một Channel kết nối tới một Instrument thông qua khái niệm cổng (port) trên instrument.
- **Complex:** Tài nguyên dạng Complex đơn thuần là tập hợp những tài nguyên Instrument được kết nối với nhau bởi các Channel.

3.3 Đàm phán với một AO:

Khi thực hiện đàm phán với AO, client trước tiên sẽ tạo ra một đối tượng tài nguyên mẫu với cấu trúc thích hợp và gán cho các thuộc tính của đối tượng này:

- Một giá trị xác định.
- Giá trị cho biết thuộc tính đó không cần thiết (don't care value).
- Giá trị đại diện bởi tên biến. Các giá trị được gán bởi tên biến sẽ được tham chiếu bởi bảng giá trị. Ngoài ra, client cũng có thể đặt thêm các ràng buộc cơ bản lên các biến.

Giá trị xác định (constrant) được client gán vào thuộc tính để gửi đến AO khi client yêu cầu tài nguyên với những thuộc tính cố định. Giá trị "don't care" để chỉ những thuộc tính mà đối với client là không quan trọng. Còn những giá trị biến là những giá trị đàm phán. AO sau khi nhận được đối tượng tài nguyên mẫu sẽ trả về lại cho client với những thuộc tính biến đã được gán giá trị, đó là những giá trị mà AO có thể đáp ứng cho client. AO có thể gán cho những thuộc tính này giá trị "có thể đàm phán" (negotiable), khi đó AO sẽ cung cấp một tập các giá trị mà AO có thể đáp ứng cho thuộc tính đó. Căn cứ vào đó client sẽ chọn. Khi thuộc tính đã được chọn giá trị thì nó sẽ trở thành hằng số.

Một cách tổng quát thì một yêu cầu từ client gửi đến AO để sử dụng tài nguyên sẽ bao gồm:

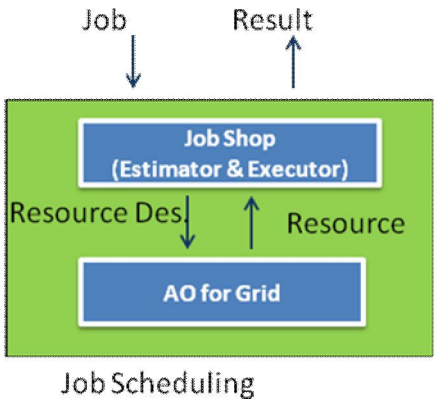
- Các thuộc tính của đối tượng tài nguyên mẫu
- Các ràng buộc trên các thuộc tính
- Kiểu đàm phán (Negotiation style), bao gồm:
  - Immediate: AO cung cấp tài nguyên cho client ngay lập tức
  - Pending: AO cần đàm phán thêm để lấy thông tin cho tài nguyên
  - Confirmation: AO và client đã đàm phán xong và sẵn sang cung cấp tài nguyên theo kết quả phiên đàm phán
  - Cancel: Hủy bỏ kết quả của phiên đàm phán.

Phương thức nhận tài nguyên (Pickup Approach), như AO sẽ thông báo cho client khi tài nguyên sẵn sang, hay client sẽ kết nối đến AO để nhận tài nguyên vào một thời điểm định trước.

- Chứng thực: dung để chứng thực trong trường hợp yêu cầu
- Chi phí chấp nhận để sử dụng tài nguyên (Bid): chi phí cao nhất mà client có thể chấp nhận để sử dụng tài nguyên.
- Định danh của phiên đàm phán (Negotiation ID): sử dụng như một session ID để xác định khi tài nguyên sẵn sang. Khi client nhận tài nguyên từ AO, client sẽ phải cung cấp định danh của phiên đàm phán để nhận tài nguyên.

3.4 Job Shops

Trong mô hình AO vừa được đề cập ở trên, mỗi AO thường là một đại diên của một tài nguyên riêng rẽ, do đó so với các mô hình khác, AO chưa có được chức năng phân chia công việc với những tác vụ lớn. Khái niệm Job Shop được đưa ra để giải quyết vấn đề này.



Job Shob

Job Shop nhận vào một tác vụ phức hợp và nhiệm vụ của nó là phân chia tác vụ này thành những tác vụ con nhỏ hơn và lên lịch để thực hiện những tác vụ này (Job Scheduling). Job Shop có 2 thành phần chính:

- **Estimator**, có nhiệm vụ:
  - Nhận tác vụ từ client, xác định khoảng thời gian sớm nhất tác vụ có thể được hoàn thành.
  - Giao tiếp với các AO để yêu cầu tài nguyên
  - Lên lịch thực hiện các tác vụ nhỏ khi mà tài nguyên đã sẵn sang từ các AO.
- **Executor**, có nhiệm vụ:
  - Nhận tài nguyên từ các AO
  - Chuẩn bị môi trường cần thiết để thực hiện các tác vụ nhỏ
  - Khởi động các tác vụ nhỏ khi tài nguyên đã sẵn sang và thu thập kết quả
  - Dọn dẹp và giải phóng tài nguyên khi tác vụ kết thúc
  - Trả kết quả cuối cùng về cho client.

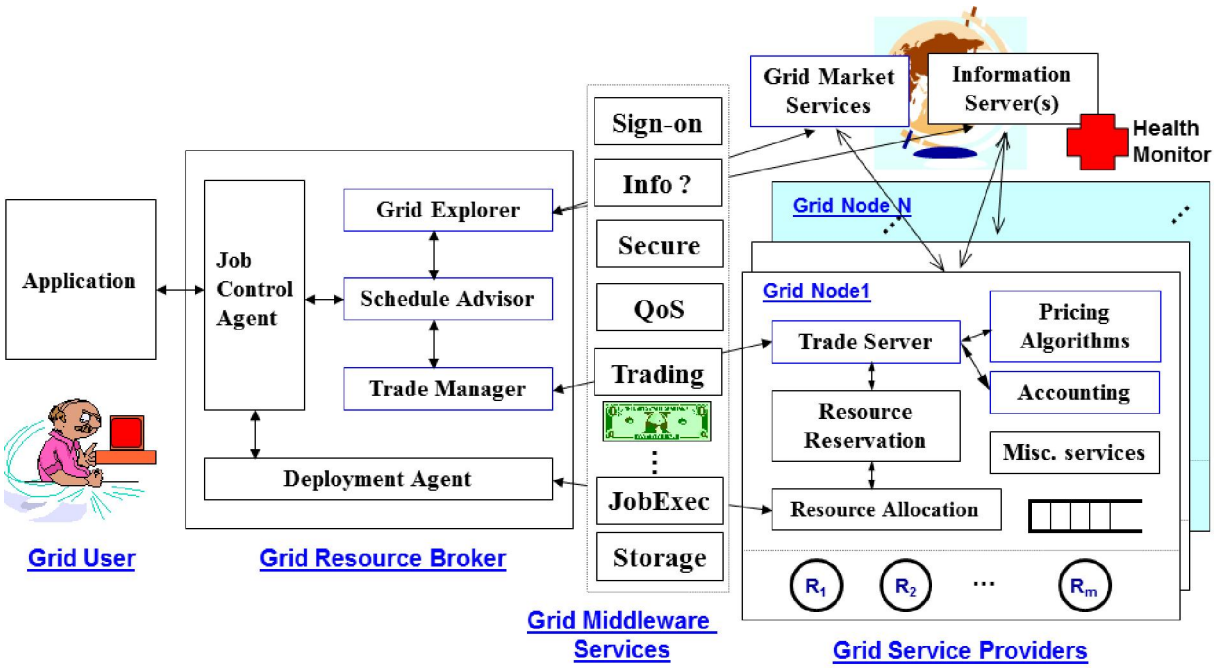
3.5 Tóm tắt

Mô hình đề nghị AO được trình bày trong bài báo chỉ ở mức độ là một mô hình đề xuất, nó còn rất nhiều khiếm khuyết cần được khắc phục trước khi có thể trở thành một mô hình Grid trên thực tế. Một trong những khiếm khuyết này là mô hình chưa đưa ra được cách thức để một client khi cần sử dụng tài nguyên có thể tìm kiếm ra nguồn tài nguyên đó hiện đang được quản lý bởi AO nào.

4 Mô hình Computational Market/Economy

Tài nguyên trên môi trường tính toán lưới nằm trên nhiều vùng địa lý khác nhau, mỗi thành phần có bộ phận quản lý tài nguyên riêng, các quy định riêng của nó, và có thể lấy giá khác nhau cho các nhu cầu khác nhau từ thị trường tính toán. Thực tế đòi hỏi phải có một mô hình hướng thị trường để những tổ chức sở hữu tài nguyên có thể tối đa hóa lợi nhuận, đồng thời giúp người dùng giảm thiểu chi phí mà vẫn đạt được kết quả như mong đợi. Mô hình này phải có các công cụ và dịch vụ giúp người dùng định nghĩa ra chất lượng dịch vụ họ cần; giúp phía quản lý tài nguyên có khả năng thích ứng cao với những thay đổi theo từng thời điểm.

Mô hình dạng này mang cả hai mô hình, phân cấp và AO.

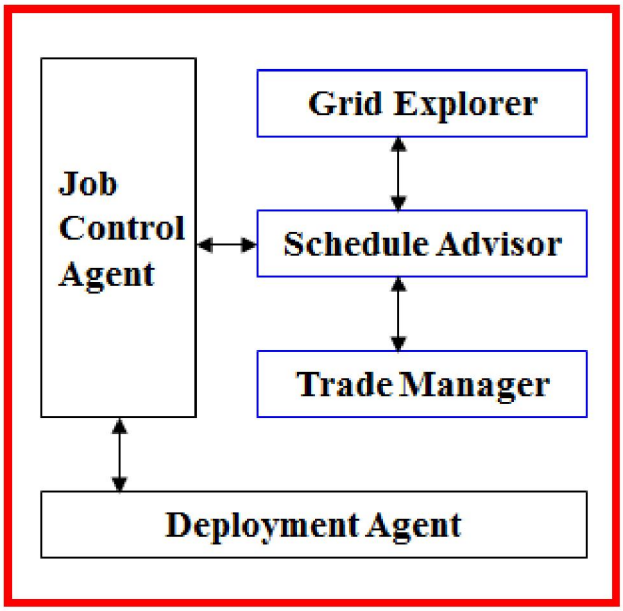


Một sơ đồ có thể thực hiện theo mô hình hướng thị trường

Những thành phần chính:

- User Applications (sequential, parametric, parallel, or collaborative applications)
- Grid Resource Broker (môi giới tài nguyên lưới - Super/Global/Meta Scheduler)
- Grid Middleware Services
- Grid Service Providers

4.1 Grid Resource Broker (Môi giới tài nguyên mạng lưới - GRB)



Grid Resource Broker

GRB đóng vai trò như thành phần trung gian giữa người dùng và các tài nguyên lưới, dựa vào các dịch vụ middleware. Chịu trách nhiệm:

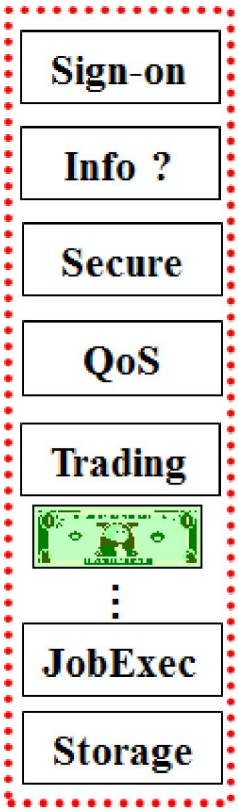
- Phát hiện tài nguyên, chọn tài nguyên, kết nối các phần mềm ứng dụng, dữ liệu và các tài nguyên phần cứng
- Khởi động các quá trình tính toán, phù hợp theo sự thay đổi tài nguyên trên mạng lưới, và trình bày mạng lưới với người dùnh như một thể thống nhất.

Các thành phần khi của một resource broker gồm:

- **Job Control Agent** (JCA): là thành phần trung tâm thống nhất, chịu trách nhiệm :
  - Luân chuyển job đi qua các thành phần của hệ thống,
  - Trông coi quá trình lập lịch, là đơn vị thực hiện tạo ra các jobs,
  - Theo dõi tình trạng của một job
  - Giao tiếp với khách hang/người dung , scheduler adviser, dispatcher
- **Schedule Advisor** (Scheduler): có trách nhiệm phát hiện tài nguyên (dựa vào grid explorer), chọn tài nguyên và phân task. Chức năng chính của nó là chọn các tài nguyên khới với yêu cầu người dùng như thoải mãn việc cho ra kết quả đúng thời hạn và tối thiểu hóa chi phí tính toán.
- **Grid Explorer**: có trách nhiệm phát hiện ra các tài nguyên bằng cách giao tiếp với grid-information server (máy chủ chứa thông tin dịch vụ) và xác định danh sách các máy tính đã được xác thực, và theo dõi thông tin tình trạng của tài nguyên.
- **Trade manager** (TM): làm việc theo hướng của thuật toán chọn tài nguyên để xác định chi phí sử dụng tài nguyên. Giao tiếp với trade server và thương lượng để sử dụng các dịch vụ với giá thấp. Nó có thể tìm giá dịch vụ từ máy chủ thông tin dịch vụ nếu chủ sở hữu tài nguyên có đăng.

- **Deployment Agent**: trách nhiệm cho việc kích hoạt các task trên những thành phần tài nguyên đã được chọn như chỉ thị của Scheduler. Nó định thời cập nhật tình trạng thực hiện task tới JCA

4.2 Grid Middleware Services (Dịch vụ trung gian tính toán lưới)



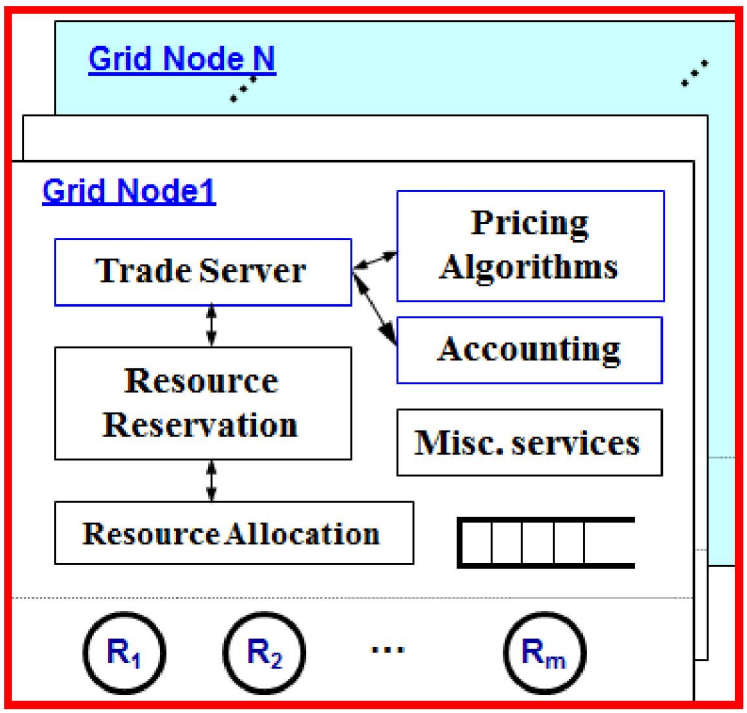
Grid Middleware Services

Grid Middleware Service cung cấp các dịch vụ giúp cho quá trình kết nối giữa người dùng và các tài nguyên từ xa, thông qua môi giới tài nguyên (GRB) hoặc các ứng dụng có thể chạy trên grid. Nó cung cấp các dịch vụ lõi như quản lý tiến trình từ xa, phân bổ tài nguyên, truy cập dữ liệu, thông tin, dịch vụ an toàn, chứng thực, và chất lượng dịch vụ như đặt trước dịch vụ để đảm bảo sự khả dụng của tài nguyên, và chào giá để tối thiểu chi phí.

Các thành phần chính của Grid Middleware services là:

- **Core services**: bao gồm remote process management, cấp phát tài nguyên, truy xuất dữ liệu, dịch vụ thông tin, bảo mật, định danh người dùng.
- **Quality of Service** (QoS): đảm bảo chất lượng tối thiểu cho các dịch vụ

4.3 Grid service providers (Dịch vụ cung cấp tính toán lưới)



Grid Service Providers

Grid Service Provider đóng vai trò của các nhà cung cấp dịch vụ tính toán lưới vụ (chủ sở hữu của các tài nguyên). Nơi để các GRBs và các GSPs thỏa thuận chi phí cho việc sử dụng tài nguyên mà các GSPs có là Grid Market Directory (GMD).

Các thành phần của một GSP là:

- **Local resource managers:** Các bộ quản lý tài nguyên nội bộ có trách nhiệm quản lý và lập lịch tính toán trên tất cả các tài nguyên nội bộ như máy trạm hoặc cluster. Chúng có thể có trách nhiệm cung cấp truy cập tới thiết bị lưu trữ, cơ sở dữ liệu, và các thiết bị khoa học, như kính viễn vọng…
- **Trade server** (TS) là một tác nhân quản lý tài nguyên, đàm phán với người dùng và bán các truy xuất tới tài nguyên, hướng tới việc việc xử dụng tối đa tài nguyên và lợi nhuận cho các tổ chức sở hữu (kiếm nhiều tiền nhất có thể) Nó thăm dò các thuật toán tính giá định nghĩa bởi người dùng trong quá trình đàm phán. Và hướng hệ thống kế toán ghi lại sự sử dụng tài nguyên
- **Pricing algorithms/Methods**: quyết định giá cho các dịch vụ. Nó có thể theo một số quy định khác nhau để tối đa hóa lợi nhuận. Cho phép nhưới dùng thay đổi giá dịch vụ theo thời gian
- **Accounting system**: có trách nhiệm ghi nhận lại việc sử dụng tài nguyên và tính chi phí cho người dùng dựa vào bộ phận quản lý tài nguyên giữa ResourceBroker và trade server.

4.4 Các mô hình mua bán resource

Có nhiều mô hình thỏa thuận mua bán resource, mô phỏng theo thế giới thật:

- Commodity Market Model (thị trường tự do)
- Posted Price Model (niêm yết giá khuyến mãi)
- Bargaining Model (thỏa thuận giá)
- Tendering/Contract-Net Model (đấu thầu)
- Auction Model (đấu giá)
- Bid-based Proportional Resource Sharing Model (chia theo tỉ lệ đặt cọc)
- Community/Coalition/Bartering Model (cổ đông, góp vốn)
- Monopoly and Oligopoly (độc quyền / chiếm đa số)

Chi tiết về các mô hình mua bán được trình bày bởi nhóm thuyết trình về Grid Economy, nên phần này chỉ liệt kê ngắn gọn các ý chính.

4.5 Danh sách một vài hệ thống grid áp dụng mô hình Economy / Market

SYSTEM NAME	ECONOMY MODEL	PLATFORM	REMARKS
Mariposa [8] (UC Berkeley)	Bidding (Tendering/ ContractNet). Pricing based on load and historical info.	Distributed database.	It supports budget-based query processing and storage management.
Mungi [17] (University of New South Wales)	Commodity market (renting storage space that increases as available storage runs low, forcing users to release unneeded storage.)	Storage servers.	It supports storage objects based on bank accounts from which rent is collected for the storage occupied by objects. .
Nimrod-G [1][2] (Monash University)	It supports economy models such as commodity market, spot market, and contract-net for price establishment.	A Grid of distributed computers (PCs, WS, and clusters)	It supports deadline and budget constrained scheduling algorithms for executing task-farming applications on large scale distributed systems depending on their cost, power, and availability and users quality of service requirements.
Popcorn [21] (Hebrew University)	Auction. (Highest bidder gets access to resource and it transfers credits from buyer to the seller account.)	Web browsers. (*Popcorn parallel code* runs within a browser of CPU cycles seller.)	Popcorn API-based parallel applications need to specify a budget for processing each of its modules.
Java Market [18] (John Hopkins University)	QoS based computational market. (The resource owner receives f(j, t) award for completing f in time t.)	Web browsers. (JavaMarket runs *standard Java Applets* within a browser).	One can sell CPU cycles by pointing Java-enabled browser to Portal & allow execution of Applets.
Enhanced MOSIX [19] (Hebrew Uni., Israel)	Commodity market (resource cost of each node is known)	Clusters of computers (Linux PCs)	It supports process migration such that overall cost of job execution is kept low.
JaWS [30] (University of Crete)	Bidding (Tendering)	Web browsers	It is similar to Popcorn.
Xenoservers [31] (University of Cambridge)	Bidding (Proportional resource sharing)	Single computer	Accounted execution of untrusted code.
D'Agents [32] (Dartmouth College)	Bidding (Proportional resource sharing)	Single computer or Mobile Agents	Agents bid function is proportional to benefit.
Rexec/Anemone [22] (UC Berkeley)	Bidding/Auction (for proportional resource sharing)	Clusters (A market-based Cluster Batch Queue System)	Users assign utility value to their application and system allocates resources proportionally.
Mojo Nation [24] (Autonomous Zone Industries, CA)	A Credit-based partnership and/or bartering model. (Contributors earn credits by sharing storage and spend them when required)	Network storage.	It is a content-sharing community network. It combines marketplace and bartering approach for file/resource sharing.
Spawn [20] (Xerox PARC)	Second-price Auction (uses sponsorship model for funding money to each task depending on some requirements)	Network on workstations. Each WS executes a single task per time slice	It supports execution of concurrent program expressed in the form of hierarchy of processes that expand and shrink size depending on the resource cost.
Supercomputing center [33] (Computing Services for Academic Research at the University of Manchester)	Commodity market and priority-based model (they charge for CPU, memory, storage, and human support services)	MPPs, Crays, and Clusters, and Storage servers.	Any application can use this service and QoS is proportional to user priority and scheduling mechanisms.

4.6 Một số đánh giá:

Các dịch vụ cung cấp bởi TradeServer có thể truy xuất từ hoặc cung cấp bởi các Information server. Trong trường hợp này, một Trade Manager hoặc Broker có thể truy xuất trực tiếp vào Information server, Trade Manager có thể dùng giá được niêm yết hoặc mời những biểu giá cạnh tranh hoặc đấu giá để chọn ra các tài nguyên thỏa mãn nhu cầu người dùng.

Từ các thảo luận trên, có thể thấy có nhiều cách khác nhau để xác định hoặc biết chi phí truy xuất. Sơ đồ ví dụ đã cung cấp là một trong những lựa chọn có thể thực hiện cho mô hình hướng thị trường, và nó có thể khác nhau dựa vào giao thức trao đổi thông tin như ở thế giới thực.

Những hệ thống lưới khác nhau có thể theo nhiều định hướng khác nhau để hoạt động, và nó sẽ rất hữu ích nếu những hệ thống này liên thông . Sự liên thông này có thể được phát triển từ các cộng đồng lưới hoặc các tổ chức chuẩn hóa như GF và eGrid.

Tài liệu tham khảo

1. Ian Foster, Carl Keselman, The GRID 2 – Blueprint for a New Computing Infrastructure
2. Rajkumar Buyya, Steve Chapin, and David DiNucci, Architectural Models for Resource Management in the Grid, USA, 2000
3. Rajkumar Buyya, David Abramson, and Jonathan Giddy, An Economy Driven Resource Management Architecture for Global Computational Power Grids.
4. Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger, Economic Models for Resource Management and Scheduling in Grid Computing.
5. David Abramson, Rajkumar Buyya, Jon Giddy. School of Computer Science and Software Engineering. Monash University, Melbourne, Australia. Nimrod/G GRID Resource Broker and Computational Economy.
6. Klaus Krauter1, Rajkumar Buyya and Muthucumaru Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing.