HỆ THỐNG TÍNH TOÁN HIỆU NĂNG CAO SUPERNODE II
A HIGH PERFORMANCE COMPUTING SYSTEM: SUPERNODE II

Thoại Nam[1], Trần Nguyễn Hoàng Huy[1], Đoàn Việt Hưng[1],
Trần Ngọc Minh[1], Nguyễn Cao Đạt[1] và Đặng Tuấn Nghĩa[2]

[1]Khoa Công Nghệ Thông Tin, Đại học Bách Khoa, Tp. Hồ Chí Minh, Việt Nam
[2]Trung tâm Điện Toán, Đại học Bách Khoa, Tp. Hồ Chí Minh, Việt Nam
nam@dit.hcmut.edu.vn

BẢN TÓM TẮT

Nhu cầu tính toán trong các lĩnh vực khoa học và công nghệ đã thúc đẩy việc xây dựng các hệ thống tính toán hiệu năng cao. Bên cạnh đó, sự phát triển của công nghệ mạng tốc độ cao kết hợp với việc sử dụng các máy trạm tính toán rẻ tiền cho phép xây dựng các hệ máy tính mạnh với giá thành hạ. Supernode II là một hệ thống tính toán dạng như vậy. Báo cáo trình bày tổng quan về phần mềm hệ thống xây dựng trên Supernode II. Các phần cơ bản trong phần mềm hệ thống này là quản lý tài nguyên, định thời, thực thi công việc, quản lý người dùng và giám sát hệ thống. Phần mềm hệ thống cho phép sử dụng tài nguyên trên Supernode II một cách hiệu quả.

ABSTRACT

High Performance Computing (HPC) is a key solution to provide computational power to solve grand and challenging problems. There has been a dramatic shift within the HPC field from supercomputers to clusters for lowering the ratio between cost and performance. This trend was set in motion by improvements of the two commodity components: CPUs designed for PCs and high-speed local networks. Supernode II is a system built on PC-based cluster technology that provides cost-effective computing power for scientific applications. This paper will give an overview of the Supernode II's system software. Consisting of five main modules: resource management, job scheduling, job execution, user management and monitoring, this software enables effective resource utilization.

1. GIỚI THIỆU

Sự phát triển của công nghệ cho phép tốc độ tính toán của bộ xử lý ngày càng tăng. Tuy vậy, có nhiều ứng dụng trong lĩnh vực khoa học và công nghệ đòi hỏi thời gian tính toán từ vài ngày đến vài tháng hay nhiều hơn nữa, cho dù tận dụng hết khả năng của một bộ xử lý. Do thời gian tính toán quá lớn, nên có thể xảy ra tình trạng kết quả tính toán không sử dụng được, ví dụ như bài toán dự báo thời tiết. Xử lý song song là một giải pháp nhằm giải quyết vấn đề trên. Sử dụng nhiều bộ vi xử lý cùng lúc để giải quyết một bài toán có thể rút ngắn thời gian tính toán, đáp ứng được nhu cầu trong thực tế. Một môi trường tính toán có thể cung cấp một khả năng tính toán mạnh để giải quyết một bài toán lớn trong một thời gian ngắn được gọi là môi trường tính toán hiệu năng cao (High Performance Computing).

Nhiều mô hình máy tính đã được nghiên cứu và phát triển nhằm phục vụ cho việc xử lý song song. Trong đó, hai mô hình phổ biến, được sử dụng nhiều trong thực tế là mô hình máy tính đa xử lý và mô hình cluster. Các máy tính đa xử lý là các máy tính có nhiều bộ vi xử lý trên cùng bo mạch và chia sẻ bộ nhớ. Mô hình cluster là sự kết nối các máy tính đơn lại với nhau, không chia sẽ bộ nhớ.

Ngày nay, khi máy tính cá nhân trở nên phổ biến và tốc độ truyền thông mạng được tăng lên rất nhiều lần thì mô hình cluster tỏ ra

có nhiều ưu điểm. Có thể kể ra một số ưu điểm của mô hình cluster như: khả năng tận dụng nguồn tài nguyên sẵn có, tính linh động, khả năng mở rộng cao, dễ xây dựng… Một ưu điểm nổi bậc của mô hình cluster là tính kinh tế; tỉ số giữa giá thành và hiệu suất của hệ thống cluster thấp hơn nhiều so với một máy tính đa xử lý có khả năng tính toán tương đương.

Từ nhu cầu thực tế của trường Đại học Bách Khoa TP. HCM, một hệ thống tính toán hiệu năng cao dạng PC-based cluster có tên Supernode II đang được xây dựng. Thiết kế của Supernode II nhằm đáp ứng nhu cầu tính toán đa dạng từ các ứng dụng tuần tự đến các ứng dụng song song sử dụng MPI và PVM. Vấn đề khó khăn trong việc phát triển các hệ thống PC-based clusters nói chung và Supernode II nói riêng chính là phần mềm hệ thống. Phần này sẽ quyết định hệ thống Supernode II chạy hiệu quả hay không.

Các phần mềm hệ thống cho các hệ thống cluster hiện có hoặc là ở dạng thương mại với chi phí rất cao, hoặc được thiết kế, hiện thực theo từng hệ thống cụ thể và không open-source hoàn toàn. Chính vì vậy để đáp ứng nhu cầu tính toán thì cần xây dựng một phần mềm phù hợp với hệ thống Supernode II. Việc xây dựng được một hệ thống phần mềm như vậy giúp khai thác tối đa hiệu suất của phần cứng và có thể phát triển theo nhiều hướng khác nhau trong tương lai mà không bị những hạn chế như khi sử dụng phần mềm có sẵn.

Một phần mềm hệ thống dùng cho hệ thống tính toán hiệu năng cao cần phải thực hiện được những nhiệm vụ sau [8]:

- Quản lý tài nguyên và định thời: quản lý toàn bộ tài nguyên trên hệ thống và phân bổ các tài nguyên hợp lý nhằm phục vụ công việc của người sử dụng.
- Giám sát hệ thống: cung cấp cho người sử dụng hệ thống khả năng giám sát công việc của mình đang được thực thi trong hệ thống. Cung cấp những thông tin về trạng thái của các công việc và tài nguyên cho người quản trị.
- Quản lý người sử dụng và tính phí (user management & billing): Do cluster thường là một hệ thống gồm nhiều người dùng nên cần có cơ chế để quản lý người sử dụng, giám sát thông tin về sử dụng tài nguyên của người dùng.
- Môi trường để biên dịch, kiểm tra chương trình.

Các phần tiếp theo của bài báo được trình bày như sau: Phần 2 giới thiệu, đánh giá sơ lược một số phần mềm quản lý cluster. Phần 3 trình bày tổng quan kiến trúc và những thành phần của phần mềm hệ thống cho Supernode II. Những hướng phát triển của đề tài được trình bày trong Phần 4. Cuối cùng là phần kết luận.

2. CÁC CÔNG VIỆC LIÊN QUAN

2.1. Hệ thống CONDOR

Condor [11] được phát triển tại đại học Wisconsin-Madison. Condor được xây dựng với mục tiêu tận dụng những tài nguyên sẵn có để xây dựng một hệ thống tính toán lớn. Condor có khả năng kết hợp giữa các nút tính toán chuyên biệt (dedicated nodes), như các nút tính toán của một cluster, với các máy tính cá nhân có thời gian rảnh (optional workstation). Chính kiến trúc linh động này cho phép Condor có khả năng mở rộng rất lớn và hướng tới xây dựng mô hình tính toán lưới (Grid computing).

Chức năng chính của Condor là quản lý tài nguyên, phân bổ công việc lên các tài nguyên một cách hợp lý nhất. Khi người dùng đưa ra một công việc, Condor sẽ tìm một máy sẵn sàng để thực thi công việc đó. Khi phát hiện một máy không thể tiếp tục thực hiện công việc, Condor có thể ngưng, lưu ảnh công việc và di dời sang một máy tính đang sẵn sàng khác. Việc lưu ảnh công việc còn cung cấp khả năng chịu lỗi cho ứng dụng.

Ngoài ra, Condor còn có cơ chế phân bố tài nguyên dựa trên độ ưu tiên người dùng (user priorities) và độ ưu tiên công việc (job priorities). Cơ chế dựa trên các độ ưu tiên này sẽ đảm bảo quá trình phân phối tài nguyên công bằng giữa những người dùng cũng như giữa các công việc. Sử dụng Condor, người quản trị dễ dàng giám sát quá trình thực thi các công việc và đưa ra những thay đổi về chiến lược hay những biện pháp xử lý công việc.

Condor không yêu cầu bất kì sự thay đổi nào về source code của chương trình khi thực thi trên Condor. Người sử dụng có thể truy cập và thực thi công việc trên Condor từ xa. Condor linh động sử dụng nhiều cơ chế khác nhau để các công việc có thể truy xuất các file dữ liệu. Các cơ chế này bao gồm: gọi hàm hệ thống từ xa (remote system call), hệ thống file

chung (shared file system) hay cơ chế chuyển file (file transfer).

Condor hỗ trợ nhiều mô hình công việc khác nhau như các công việc tuần tự, PVM và MPI. Bên cạnh Linux, Condor có thể hỗ trợ nhiều hệ điều hành khác trong dòng Unix cũng như hệ điều hành Windows NT. Trong một nhóm máy (pool) có thể có nhiều hệ điều hành khác nhau và một công việc thực thi trong một hệ điều hành này có thể được đăng ký (submit) từ một máy với hệ điều hành khác. Cơ chế kết nhóm (flocking) cho phép nhiều nhóm có thể được kết nối lại với nhau. Mỗi nhóm có thể có những chiến lược quản lý tài nguyên khác nhau và công việc có thể được thực thi trên nhiều nhóm khác nhau. Thông qua Globus Toolkit, Condor có thể gởi công việc đến một hệ thống được quản lý bởi một hệ thống quản lý tài nguyên khác cũng như nhận công việc từ hệ thống khác gởi tới. Điều này giúp Condor dễ dàng thích nghi với mô hình tính toán lưới.

2.2. Computing Center Software

Computing Center Software (CCS) [10] là một hệ thống quản lý tài nguyên cho các máy tính song song hiệu năng cao, được nghiên cứu bởi trung tâm Paderborn và được đưa vào hoạt động từ năm 1992 trên nhiều hệ thống clusters cũng như các hệ thống tính toán song song lớn. CCS cung cấp khả năng định thời các công việc bó (batch jobs) và tương tác độc lập với phần cứng. Việc định thời của CCS dựa trên các yếu tố như sự đặt chỗ (reservation), hạn chót (deadline), các hạn chế về thời gian như phân biệt giữa ngày và đêm, các hạn chế về số lượng nút. Ngoài ra CCS còn cung cấp khả năng phân chia các tài nguyên độc quyền và không độc quyền, các giao diện mở để giao tiếp với các hệ thống quản lý tài nguyên khác. CCS có độ tin cậy cao như tự động khởi tạo lại khi các quá trình chạy ngầm (daemon) bị chết, khả năng đề kháng lỗi trong trường hợp đứt mạng.

2.3. Portable Batch System

Portable Batch System (PBS) [12] là một hệ thống bó linh hoạt quản lý tải và hàng đợi tuân theo chuẩn POSIX, được phát triển và duy trì bởi Veridian Systems. PBS hoạt động trên các môi trường mạng UNIX, bao gồm các cluster bất đồng nhất, các siêu máy tính và các hệ thống song song lớn. Dự án này được khởi động để tạo ra một hệ thống xử lý bó linh hoạt và có khả năng mở rộng để đáp ứng những yêu cầu của các mạng tính toán bất đồng nhất. Mục đích của PBS là để cung cấp các điều khiển cho việc khởi tạo hay định thời các công việc bó. Bộ định thời mặc định trong PBS là FIFO dùng để tối ưu hiệu suất của CPU. Bộ định thời này sẽ duyệt qua danh sách hàng đợi công việc và khởi động bất kỳ công việc nào phù hợp với các tài nguyên đã sẵn sàng. Tuy nhiên, định thời theo phương pháp này sẽ khiến các công việc lớn không có cơ hội được thực thi. Để khắc phục, bộ định thời này sử dụng cơ chế "starving jobs". Với cơ chế này, công việc khi ở trong hàng đợi quá một ngưỡng thời gian xác định sẽ được tăng độ ưu tiên lên, để nó được đưa vào thực thi ngay khi có đủ tài nguyên. Phương pháp này có thể thực hiện tốt trong vài trường hợp, nhưng trong nhiều tình huống không mang lại kết quả mong muốn [1]. Tuy nhiên, PBS cho phép thay thế bộ định thời mặc định của nó. Một số bộ định thời có lựa chọn, có thể sử dụng với PBS đã ra đời. Maui [13] và Libra [3] là những bộ định thời như vậy, thích hợp cho các môi trường tính toán hiệu năng cao.

Maui sử dụng các chính sách định thời tổng hợp để tối ưu việc tận dụng tài nguyên và thời gian hoàn thành công việc. Maui cung cấp khả năng quản trị về tài nguyên và tải công việc, cho phép cấu hình trong việc sắp xếp độ ưu tiên, định thời, cấp phát, sự công bằng, và các chính sách đặt chỗ trước cho công việc. Maui cũng cho phép việc đặt chỗ trước để điều khiển chính xác khi nào tài nguyên được sử dụng.

Mục tiêu của bộ định thời Libra là cố gắng thỏa mãn tối đa yêu cầu của người sử dụng. Vì vậy các chi tiết công việc được trình lên bởi người sử dụng bao gồm: ngân sách được cấp và hạn chót hoàn thành công việc. Bộ định thời Libra cho phép tối ưu độ lợi của CPU trong khi vẫn đảm bảo những yêu cầu ràng buộc về chất lượng dịch vụ của người sử dụng.

Nhờ vào tính ổn định, linh hoạt, khả năng cho phép thay thế các bộ định thời khác nhau để thỏa mãn các chính sách của từng đơn vị, đặc biệt là tính chất mã nguồn mở giúp cho PBS trở thành một hệ thống phổ biến và được sử dụng rộng rãi.

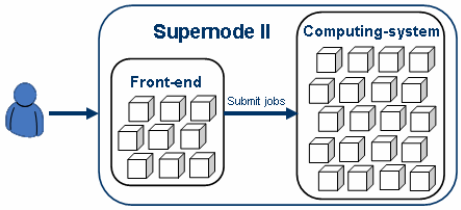
Tuy nhiên, vì PBS là một hệ thống phi thương mại nên việc tính phí sử dụng cũng như quản lý người dùng không được phát triển trên hệ thống này.

3. PHẦN MỀM QUẢN LÍ CHO HỆ THỐNG SUPERNODE II

Hệ thống Supernode II được xây dựng gồm có 64 nút, mỗi nút có cầu hình gồm:

- 2 CPU Xeon 2,4GHz, 512KB cache
- 2 GB ECC DDRam
- HDD 40GB (7200rpm)

Các nút được kết nối với nhau theo kiểu bus chung (shared-bus) thông qua 2 Ethernet Switch 1Gigabit và 100Mb.



Hình 1. Sự phân chia các thành phần của hệ thống.

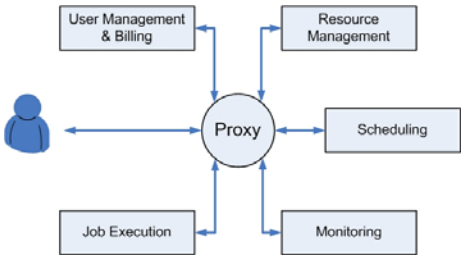
Các nút tính toán được phân chia làm hai phần Front-end và Computing system như trong Hình 1. Phần Front-end (gồm 16 nút) cung cấp ngõ vào để người dùng đăng nhập và thực thi ứng dụng trên hệ thống. Người dùng có thể biên dịch và chạy thử chương trình trên các máy tính của Front-end để kiểm tra tính chính xác và khả năng tương thích của ứng dụng với hệ thống. Các máy tính thật sự thực thi công việc nằm ở phần Computing system (gồm 48 nút). Phần này hoàn toàn trong suốt đối với người dùng và được quản lý bởi phần mềm quản lý Supernode II.

Lớp ứng dụng
Lớp phần mềm hệ thống quản lý cluster
Hệ điều hành
Phần cứng máy tính

Hình 2. Sơ đồ phân lớp.

Phần mềm hệ thống quản lý Supernode II là lớp phần mềm nằm giữa lớp ứng dụng của người dùng và hệ điều hành trên mỗi máy đơn. Lớp phần mềm này sẽ cung cấp một cái nhìn trong suốt về hệ thống cho người dùng ở lớp ứng dụng. Người dùng thực thi, phát triển các ứng dụng dựa trên lớp phần mềm hệ thống mà không cần quan tâm đến cấu trúc các máy tính bên dưới.

Cấu trúc chung của phần mềm quản lý hệ thống Supernode II gồm có những thành phần như trong Hình 3. Proxy đóng vai trò trung gian trong việc giao tiếp giữa các module với nhau. Proxy cũng đóng vai trò là trung gian giữa người sử dụng và hệ thống. Chi tiết từng chức năng của của các bộ phận khác được trình bày tiếp trong các phần kế tiếp.



Hình 3. Cấu trúc của phần mềm hệ thống.

3.1. Quản lý tài nguyên (Resource Management)

3.1.1. Chức năng

Bộ quản lý tài nguyên có nhiệm vụ quản lý tất cả tài nguyên tính toán, bao gồm CPU, bộ nhớ, đĩa cứng, xác định vị trí và cấp phát tài nguyên cho hệ thống, cho phép tài nguyên mới gia nhập, tài nguyên hiện hành rời khỏi hệ thống, phát hiện những sự cố hư hỏng tài nguyên và cung cấp thông tin tài nguyên cho bộ định thời. Ngoài ra, bộ quản lý tài nguyên còn có các nhiệm vụ như chứng thực, tạo và di trú các quá trình.

3.1.2. Kết quả đạt được

Các chức năng đã được hiện thực của bộ quản lý tài nguyên:

- Resource Controller: cung cấp cho người quản trị chức năng thêm vào và xóa bỏ các tài nguyên, nhận yêu cầu đăng ký từ các nút tính toán khi chúng được khởi động.
- Resource Info Controller: quản lý thông tin tài nguyên của tất cả các nút tính toán. Nhận yêu cầu lấy thông tin tài nguyên từ bộ

định thời và trả về thông tin tài nguyên hiện tại của hệ thống.

- Resource Allocation Controller: nhận yêu cầu thực thi công việc từ bộ định thời, phân công việc xuống các nút tính toán để thực thi, cập nhật lại thông tin tài nguyên.
- Job Execution Controller: quản lý quá trình thực thi của công việc, cập nhật thông tin thực thi của công việc về cho bộ định thời. Nhận yêu cầu xóa và tạm dừng công việc từ bộ định thời, dừng sự thực thi của công việc, trả lại các tài nguyên công việc chiếm giữ, cập nhật lại tài nguyên hệ thống, trả kết quả về cho bộ định thời.
- Resource Maintainance Controller: gửi tín hiệu đến các nút tính toán một cách định kỳ để kiểm tra xem các nút có hoạt động không. Cập nhật lại các tài nguyên nếu có nút chết (không hoạt động, bị đứt mạng…). Nếu nút chết khi đang thực thi công việc thì phải thông báo về cho bộ định thời để có thể định thời lại công việc.

3.2. Định thời (Scheduling)

3.2.1. Chức năng

Bộ định thời có nhiệm vụ chính là quyết định thời điểm và thứ tự xử lý các công việc của người dùng chứa trong hàng đợi. Bộ định thời cũng xác định tài nguyên nào được dùng để thực thi mỗi công việc. Trong quá trình định thời, bộ định thời lấy thông tin tài nguyên từ bộ quản lý tài nguyên.

3.2.2. Các phương pháp định thời

Hầu hết những phương pháp định thời rơi vào một trong hai loại là time-sharing và space-sharing [9]. Time-sharing tỏ ra có ưu thế hơn khi có sự khác biệt nhiều về thời gian thực thi giữa những quá trình của cùng một công việc, hoặc khi thời gian thực thi của các công việc khác biệt lớn; ví dụ trong những hệ thống hỗ trợ cả công việc dạng bó và công việc tương tác (interactive job). Ngược lại, space-sharing không bị phí tổn do chuyển đổi ngữ cảnh. Đối với những hệ thống hỗ trợ công việc hạn cuối, space-sharing giúp cho việc ước tính thời gian thực thi của các công việc chính xác hơn. Hybrid là một phương pháp khác kết hợp ưu điểm của time-sharing và space-sharing và đã đạt được những kết quả khả quan trên thực tế [4].

Space-sharing được hiện thực bằng cách chia những tài nguyên tính toán thành những phân vùng (partition) và mỗi công việc sẽ được thực hiện trên một partition riêng biệt [9]. Space-sharing giảm thiểu tối đa phí tổn do việc chuyển đổi ngữ cảnh (context switching) giữa các công việc. Ngoài ra trong những hệ thống mà bộ nhớ bị giới hạn thì việc space-sharing dành toàn bộ bộ nhớ của cả một partition cho riêng một công việc sẽ giảm thiểu chi phí page-fault, giúp hệ thống hoạt động hiệu quả hơn.

Time-sharing sử dụng preemption để xoay vòng sử dụng các nút cho các công việc. Hai cách cơ bản khi sử dụng preemption để định thời các công việc song song là coordinated và uncoordinated. Trong cách định thời coordinated, các quá trình của cùng một công việc song song được thực thi tại cùng một thời điểm trong một quantum và cùng được đưa vào trạng thái tạm dừng tại cùng một thời điểm để nhường các nút tính toán cho công việc khác. Trong cách định thời uncoordinated thì ngược lại, mỗi quá trình của một công việc trên mỗi nút được định thời một cách độc lập mà không phụ thuộc vào các quá trình khác của cùng công việc đó. Như vậy tại một thời điểm, có một số quá trình của công việc đang thực thi, trong khi một số quá trình khác của chính công việc này ở trạng thái tạm dừng. Cách định thời coordinated chỉ hoạt động tốt đối với những ứng dụng có mức độ đồng bộ giữa các quá trình cao, ngược lại thì nó hoạt động không hiệu quả. Một nhược điểm khác của cách định thời coordinated là việc hiện thực khó khăn, đặc biệt trên những hệ thống lớn có bộ nhớ phân tán (distributed memory).

Hybrid là phương pháp định thời kết hợp giữa space-sharing và time-sharing. Ý tưởng cơ bản của hybrid là chia hệ thống thành những partition giống như space-sharing. Tuy nhiên, thay vì trên mỗi partition tại một thời điểm chỉ cho phép một công việc thực thi thì hybrid cho phép nhiều công việc cùng thực thi. Các công việc được phân bố trên cùng một partition sẽ thực thi theo kiểu xoay vòng như trong time-sharing. Một vài hiện thực theo phương pháp hybrid như hierarchical scheduling cho thấy hiệu quả đạt được tốt hơn so với space-sharing và time-sharing [7, 9].

3.3.3. Phân loại giải thuật định thời

Các giải thuật định thời được chia làm hai loại: có hạn cuối và không có hạn cuối.

Đối với công việc không hạn cuối, thông thường bộ định thời sẽ có ít nhất một hàng đợi để tiếp nhận. Nếu tài nguyên rảnh, các công việc không hạn cuối có thể thực thi ngay khi vừa được đăng ký; ngược lại, khi tài nguyên bận, những công việc này phải chờ cho đến khi có công việc kết thúc trả lại tài nguyên hoặc có thêm tài nguyên mới gia nhập vào hệ thống. Như vậy, bài toán định thời cho công việc không hạn cuối là xác định một tập công việc sẽ được thực thi khi xảy ra sự kiện có tài nguyên rảnh và những công việc này thực thi trên tài nguyên nào.

Đối với các công việc hạn cuối, thông thường bộ định thời phải dành riêng tài nguyên tính toán cho từng công việc. Số lượng nút tính toán cấp cho mỗi công việc phải bằng với yêu cầu của người dùng. Không thể thu nhỏ kích thước partition của các công việc hạn cuối, bởi vì nếu làm như vậy thì không thể dự đoán được khoảng thời gian thực thi của công việc. Điều này dẫn đến số nút tính toán của hệ thống chỉ đủ để đáp ứng được cho một số ít các công việc. Vấn đề đặt ra là làm sao để đáp ứng được nhiều công việc hạn cuối trong sự giới hạn của tài nguyên hệ thống.

3.2.4. Giải thuật định thời cho hệ thống

Giải thuật định thời của hệ thống Supernode II cho các công việc không hạn cuối dùng phương pháp space-sharing với cách phân vùng "adaptive partitioning", gồm hai phần: tính kích thước partition và lựa chọn công việc thực thi [6]. Giải thuật mở rộng phương pháp tính kích thước partition được đề nghị bởi Sivarama P. Dandamudi và Hai Yu [5] bằng cách xem xét thêm một số đặc thù của việc thực thi công việc space-sharing. Bằng cách cố gắng tạo ra những partition có kích thước bằng nhau và thích ứng với tình trạng tải hiện hành của hệ thống, phương pháp này giảm thiểu tối đa sự phân mảnh tài nguyên.

Tính kích thước partition.

Kích thước partition cho một công việc được tính theo công thức sau:

$$parsize = pmf * Max(1, ceil(\frac{total\_processors}{queue\_length + 1 + f*S}))$$

Công thức 1: Tính kích thước partition.

Trong đó:

- total_processors là số lượng nút tính toán.
- queue_length là số lượng công việc đang chờ.
- S là số lượng công việc đang thực thi.
- f là hệ số ảnh hưởng của các công việc đang thực thi, $0 \le f \le 1$.

Cách tính này cố gắng phân chia những partition bằng nhau cho tất cả các công việc đang chờ trong hệ thống. Việc dùng queue_length + 1 (thay vì queue_length) nhằm mục đích chừa một partition trống cho công việc đến trong tương lai. Sự thích ứng với tình trạng hệ thống của công thức này thể hiện ở chỗ kích thước partition bị ảnh hưởng bởi khối lượng tải của hệ thống. Khi hệ thống tải cao (queue_length và S lớn) thì công việc được phân ít tài nguyên (kích thước partition nhỏ) và ngược lại. Mức độ ảnh hưởng của những công việc đang thực thi đến kích thước partition tỉ lệ với hệ số f. Khi f = 0, sự ảnh hưởng này bị loại bỏ và kích thước partition chỉ phụ thuộc vào những công việc đang chờ. Khi f = 1, những công việc đang thực thi có ảnh hưởng ngang bằng với những công việc đang chờ, và do đó mục tiêu tạo những partition có kích thước bằng nhau đạt được tốt hơn. Báo cáo kỹ thuật của Sivarama P. Dandamudi và Hai Yu [5] cho thấy giá trị tốt nhất của f phụ thuộc vào mức độ sử dụng hệ thống, kiến trúc công việc… và f tốt nhất trong khoảng từ 0.5 đến 0.75.

Ngoài ra, kích thước partition còn phụ thuộc vào một vài yếu tố khác như đặc tính công việc và đặc tính tài nguyên của hệ thống. Không thể phân một số lượng ít tài nguyên tính toán cho một công việc song song có nhiều quá trình để đảm bảo được chất lượng phục vụ. Và cũng không thể phân một số lượng nhiều tài nguyên cho một công việc có ít quá trình vì sẽ làm giảm độ sử dụng của hệ thống. Cách tính kích thước partition chỉ phụ thuộc vào tình trạng tải hệ thống như trên sẽ không giải quyết được những vấn đề này. Một cách đơn giản để giải quyết là dùng các thông số giới hạn sau đây:

▪ MAX_PARTITION_SIZE là kích thước partition lớn nhất cho phép, nhằm đảm bảo công việc không chiếm quá nhiều tài nguyên, làm ảnh hưởng đến những công việc khác.

▪ MIN_PROCESS_PER_NODE là số lượng quá trình tối thiểu trên một nút, nhằm đảm bảo nút tính toán không bị lãng phí khả năng tính toán do số quá trình được phân quá ít.

▪ MAX_PROCESS_PER_NODE là số lượng quá trình tối đa trên một nút, tránh trường hợp công việc có nhiều process nhưng lại được cấp quá ít tài nguyên tính toán.

Lựa chọn công việc thực thi.

Tập các công việc với kích thước partition được tính như trên sẽ được lựa chọn để thực thi khi hệ thống có tài nguyên rảnh. Công việc nào được lựa chọn phụ thuộc vào nhiều yếu tố như độ ưu tiên (priority), thời gian chờ (waiting time) cũng như kích thước partition của công việc đó. Những công việc có độ ưu tiên cao, thời gian chờ lâu sẽ được xem xét trước. Bên cạnh đó, các công việc được lựa chọn sao cho tổng tài nguyên tính toán gần bằng với số lượng nút rảnh nhất để giảm thiểu tối đa tình trạng phân mảnh. Tuy nhiên việc lựa chọn đôi khi dẫn đến tình trạng trì hoãn vô hạn định của một vài công việc. Đây là vấn đề cần được phát hiện và giải quyết triệt để. Dưới đây trình bày ba chiến thuật lựa chọn công việc nhằm mục tiêu giảm thiểu tình trạng phân mảnh và một cơ chế đơn giản để giải quyết triệt để tình trạng trì hoãn vô hạn định.

1.Lặp bestfit: Đầu tiên, chọn công việc có kích thước partition lớn nhất nhỏ hơn số lượng nút rảnh. Kế tiếp, trong tập còn lại, chọn công việc có kích thước partition lớn nhất nhỏ hơn số lượng nút rảnh sau khi đã trừ đi kích thước partition của công việc đã chọn ban đầu. Lặp lại như trên cho đến khi không còn công việc hoặc số lượng tài nguyên rảnh không đủ để thực thi công việc nào nữa.

2.Lặp worstfit: Chọn các công việc theo thứ tự kích thước partition từ nhỏ đến lớn cho đến khi tổng kích thước partition vượt quá số lượng nút rảnh.

3.Vét cạn: Khi tập công việc có số lượng nhỏ thì việc xét tất cả những tập con để tìm ra lời giải tối ưu là chiến thuật hợp lý nhất. Số lượng tập con phải xét tối đa là 2^n với n là số lượng công việc cần xét. Nếu n không quá 20 thì thời gian tìm tập công việc tối ưu là chấp nhận được.

4.Cơ chế giải quyết tình trạng trì hoãn vô hạn định như sau:

- Khi thời gian chờ của một công việc vượt quá ngưỡng cho phép, công việc đó được xem xét trước tiên khi có tài nguyên rảnh. Gọi công việc đó là J và kích thước partition là p.
- Nếu đủ lượng tài nguyên rảnh thì J sẽ được thực thi ngay.
- Ngược lại, có hai trường hợp:
  4.Tồn tại một công việc K đang thực thi có kích thước partition lớn hơn của J, J sẽ chờ cho đến khi có đủ tài nguyên rảnh để thực thi. Điều này luôn luôn xảy ra vì trong trường hợp xấu nhất, khi K kết thúc, J sẽ được thực thi.
  4.Ngược lại, kích thước partition của J lớn hơn mọi công việc đang thực thi thì phải chừa lại một số lượng nút ít nhất là bằng (p – pmax) với pmax là kích thước partition lớn nhất của các công việc đang thực thi. Điều này đảm bảo khi công việc lớn nhất kết thúc thì J sẽ được thực thi.

Để tránh tình trạng một công việc lặp vô tận, MAX_EXECTUTION_TIME quy định thời gian thực thi lớn nhất mà một công việc có thể thực thi. Mọi công việc khi thời gian thực thi vượt quá ngưỡng này sẽ buộc phải dừng.

3.3. Thực thi công việc (Job Execution)

Bộ định thời sau khi chọn được công việc cần thực thi và các tài nguyên để thực thi công việc sẽ gửi thông tin này đến bộ quản lý tài nguyên để chuyển cho bộ thực thi công việc. Trách nhiệm của bộ thực thi công việc là phân công đến các nút tương ứng, thành phần thực thi công việc trên các nút này sẽ đảm nhiệm việc thực thi công việc trên các nút đó và giám sát quá trình thực thi. Trong quá trình thực thi, nếu người dùng có những thay đổi gì trên công việc thì thành phần này phải có nhiệm vụ cập nhật. Khi quá trình thực thi kết thúc thì phải trả lại tài nguyên cho hệ thống, đảm bảo không có tiến trình nào của công việc vừa kết thúc còn tiếp tục chiếm giữ tài nguyên của hệ thống.

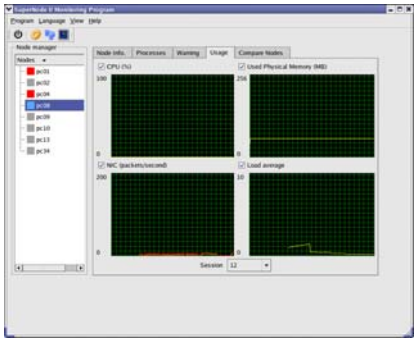
3.4. Giám sát (Monitoring)

3.4.1. Chức năng

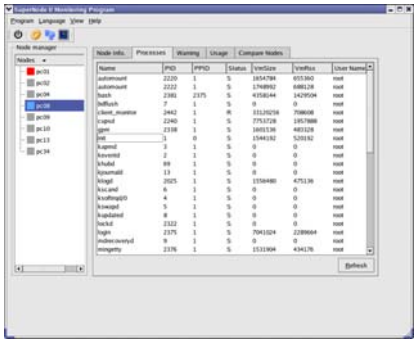
Phần mềm giám sát hệ thống cho phép người quản trị hệ thống theo dõi hoạt động hiện thời của hệ thống. Các thông tin chính cần được cung cấp là: các nút tính toán nào ở trạng thái sẵn sàng, khối lượng công việc mà mỗi nút đang đảm nhận,… Từ các thông tin này, người quản trị hệ thống sẽ đánh giá được hoạt động của hệ thống. Nếu phát hiện hệ thống bị mất cân bằng tải thì có thể can thiệp vào giải thuật định thời để phân bổ tài nguyên hợp lý hơn. Ngoài ra dựa trên thông tin ở phần giám sát, có thể biết được là hệ thống hoạt động với bao nhiêu phần trăm công suất từ đó có định hướng về việc nâng cấp hệ thống.

Về phía người sử dụng thì phần mềm giám sát cho phép người sử dụng theo dõi tiến trình thực thi các ứng dụng mình đã đăng ký.

3.4.2. Hiện thực trên Supernode II



Hình 4. Giám sát thông tin phần cứng của các nút trong hệ thống.



Hình 5. Theo dõi sự thực thi của công việc.

Module giám sát trên Supernode II đã được hiện thực và là công cụ hữu ích cho người quản trị hệ thống. Hình 4 và 5 cho thấy kết quả giám sát thông tin phần cứng của hệ thống và các công việc đang thực thi.

Một hướng phát triển module giám sát trên hệ thống Supernode II là xây dựng, tích hợp một module trợ giúp người quản trị phân tích thông tin giám sát. Khi có những biểu hiện bất thường của hệ thống, module trợ giúp sẽ phân tích và đưa ra những chỉ dẫn thích hợp cho người quản trị xử lý sự cố.

3.5. Quản lý người dùng và tính phí (User Management & Billing)

3.5.1. Chức năng của quản lý người dùng và tính phí

Quản lý người dùng trên hệ thống tính toán hiệu năng cao so với quản lý người dùng trên mạng máy tính phức tạp hơn và ở cấp độ cao hơn. Tài khoản sử dụng trên hệ thống tính toán sẽ giới hạn quyền của người sử dụng. Người sử dụng chỉ được phép đăng ký chạy ứng dụng thay vì thực thi trực tiếp ứng dụng. Việc thực thi sẽ do phần mềm hệ thống đảm nhiệm. Mặc khác do cần phải tính phí sử dụng nên sẽ có các loại tài khoản người dùng khác nhau tùy theo mức phí sử dụng. Đứng ở phía hệ thống, sẽ có rất nhiều loại tài khoản người dùng với quyền sử dụng tài nguyên khác nhau về số nút tính toán, thời điểm thực thi ứng dụng, dung lượng đĩa cứng… Những giới hạn này sẽ được xác định khi người sử dụng đăng ký tài khoản trên hệ thống.

3.5.2. Thiết kế trên Supernode II

Module quản lý người dùng trên Supernode II cho phép người sử dụng đăng ký tài khoản với hệ thống. Và với quyền đã đăng ký, người dùng có thể truy xuất vào hệ thống từ xa thông qua giao diện web hoặc kết nối từ xa bằng SSH (secure shell).

Do cách tính phí có thể thay đổi, chính vì vậy hệ thống tính phí không đưa ra một cách tính cụ thể. Thay vào đó cho phép người quản trị đặc tả phương thức tính phí. Hệ thống sẽ lưu lại các thông tin phục vụ cho tính phí, những thông tin này chủ yếu liên quan đến quá trình thực thi các ứng dụng được đăng ký bởi

người sử dụng. Từ đặc tả và thông tin đã được lưu có thể cho ra được thông tin về phí sử dụng của người dùng.

4. NHỮNG HƯỚNG NGHIÊN CỨU PHÁT TRIỂN

4.1. Tăng cường khả năng chịu lỗi (Fault-tolerant)

Các bài toán thực thi trên hệ thống tính toán hiệu năng cao thường là những bài toán có thời gian thực thi rất lớn. Khi xảy ra lỗi trên hệ thống như lỗi của phần cứng, mạng, sự cố về điện… thì phải thực thi lại chương trình từ đầu. Trong khi đó đặc điểm của các hệ thống PC-based cluster được kết nối từ những máy PC thông qua mạng nên xác suất xảy ra sự cố sẽ cao hơn so với hệ thống máy tính đa xử lý. Điều này đặt ra yêu cầu phải tăng cường khả năng chịu lỗi cho hệ thống. Một giải pháp cho vấn đề đó là sử dụng kỹ thuật lưu ảnh và khôi phục (checkpoint & recovery). Trong quá trình thực thi của ứng dụng, hệ thống sẽ định kỳ lưu lại trạng thái của chương trình, để khi có sự cố xảy ra chương trình có thể thực thi từ vị trí đã được lưu thay vì thực thi lại từ đầu. Hai loại ứng dụng được tập trung nghiên cứu để tăng cường khả năng chịu lỗi là chương trình chỉ có một quá trình và chương trình song song dạng truyền thông điệp theo chuẩn MPI, PVM. Đây là những dạng ứng dụng phổ biến hiện nay và chính là hai loại chương trình được hỗ trợ trên Supernode II.

4.1.1. Lưu ảnh và khôi phục trên một quá trình

Hiện nay đã có khá nhiều công cụ cho phép lưu ảnh và khôi phục chương trình chỉ gồm một quá trình. Tuy nhiên một trong những hạn chế của các thư viện này là chưa thực hiện tốt lưu ảnh file. Trong khi đó file thường được sử dụng để cung cấp thông tin nhập/xuất của chương trình ứng dụng. Không thực hiện tốt lưu ảnh file sẽ làm cho lớp bài toán có khả năng chịu lỗi nhỏ đi rất nhiều. Chính vì vậy phát triển cơ chế đảm bảo lưu và khôi phục lại nội dung file là một hướng ưu tiên để phát triển công cụ lưu ảnh và khôi phục trên một quá trình, tích hợp vào phần mềm hệ thống.

4.1.2. Lưu ảnh và khôi phục chương trình MPI và PVM

Đặc điểm của chương trình MPI và PVM là trạng thái của chương trình bao gồm cả trạng thái của kênh truyền. Chính vì vậy thực hiện checkpoint chương trình MPI và PVM bên cạnh việc lưu lại trạng thái của các quá trình (có thể dùng công cụ lưu ảnh trên một quá trình) cần xây dựng một phương pháp lấy được trạng thái của kênh truyền một cách nhất quán [2].

Xây dựng công cụ để lưu ảnh chương trình song song MPI và PVM đã có những kết quả bước đầu. Để hướng tới xây dựng được một công cụ lưu ảnh và khôi phục với phí tổn thấp, không gây ảnh hưởng đến sự thực thi bình thường của ứng dụng, giảm thiểu sự cần thiết phải can thiệp của người sử dụng… sẽ cần nhiều sự đầu tư nghiên cứu hơn nữa.

4.2. Cải tiến giải thuật định thời

Giải thuật định thời hiện tại chỉ dành riêng cho các công việc không hạn cuối. Tuy nhiên, trong thực tế, các ứng dụng thường đòi hỏi phải được hoàn thành trong một khoảng thời gian xác định. Vì vậy, trong tương lai giải thuật định thời sẽ được cải tiến để hỗ trợ các công việc có hạn cuối. Ngoài ra, giải thuật hiện tại chưa quan tâm đến các chính sách định thời như phân biệt thời điểm định thời, ưu tiên người sử dụng… Do đó, sắp tới nhóm sẽ tìm hiểu và định ra các chính sách định thời hợp lí để áp dụng trong việc cải tiến giải thuật định thời.

4.3. Tích hợp vào hệ thống Grid

Tính toán lưới (Grid computing) đang dần được phát triển và được xem như là giai đoạn phát triển tiếp theo của hệ thống phân bố (distributed computing). Grid cho phép chia sẻ tài nguyên tính toán (khả năng tính toán), thông tin giữa các tổ chức với nhau một cách hiệu quả, an toàn theo một giao thức chuẩn. Khả năng chia sẻ này cho phép giải quyết những bài toán tưởng chừng như không thể giải quyết được do hạn chế về khả năng tính toán và do lượng dữ liệu quá lớn. Grid cũng tận dụng những tài nguyên tính toán mà hiện nay chúng ta chưa khai thác triệt để. Đã bắt đầu hình thành khái niệm hạ tầng tính toán

lưới (Grid infrastructure), mà trong đó việc sử dụng tài nguyên tính toán, thông tin… tiện lợi như hiện nay chúng ta sử dụng mạng lưới điện.

Điểm mấu chốt của Grid là làm sao có thể liên kết những tài nguyên khác nhau vào trong hệ thống qua một chuẩn nhất định. Phần mềm hệ thống cho hệ thống tính toán hiệu năng cao Supernode II cũng cần có những thay đổi, bổ sung phù hợp để có thể gia nhập vào trong cộng đồng Grid.

5. KẾT LUẬN

Xây dựng một hệ thống cluster không chỉ đơn thuần là kết nối phần cứng các máy tính với nhau thông qua mạng có tốc độ cao. Khác biệt giữa một mạng máy tính cục bộ và một hệ thống cluster là với cluster người sử dụng chỉ thấy toàn bộ hệ thống như một máy tính có sức mạnh lớn hơn nhiều lần một máy tính đơn. Phần mềm hệ thống cho Supernode II với những chức năng được thiết kế đã cung cấp cho người sử dụng một cái nhìn nhất quán đối với hệ thống. Người sử dụng chỉ đăng ký, gửi yêu cầu thực thi công việc cho hệ thống, phần mềm hệ thống chịu trách nhiệm định thời, thực thi, giám sát công việc của người sử dụng.

Các module như quản lý tài nguyên, định thời và phân bổ quá trình, giám sát hệ thống và hệ quản lý người dùng được thiết kế hướng tới mục tiêu cao nhất là để giúp sử dụng Supernode II một cách hiệu quả. Những thiết kế này cũng có tính mở nhằm cho phép nâng cấp và tái cấu hình phù hợp với nhu cầu thực tế.

Bên cạnh việc thiết kế xây dựng phần mềm, đề tài cũng hướng đến phát triển những nghiên cứu học thuật về scheduling, fault-tolerant và grid computing.

Xin chân thành cảm ơn sự đóng góp của tất cả các thành viên khác trong nhóm nghiên cứu xây dựng Supernode II: Lê Hoàng Anh, Trương Nghĩa An , Trần Anh Dũng, Trần Vũ Ngọc Tường, Trần Đình Toàn, Nguyễn Anh Dũng và các thành viên khác.

MỤC LỤC THAM KHẢO

1. Brett Bode, David M. Halstead, Ricky Kendall and Zhou Lei. *The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters*. In 4th Annual Linux Showcase and Conference, October 2000.
2. E.N.Mootaz Elnozahy, Lorenzo Alvisi, Yi-Min Wang and David B.Johnson. *A survey of Rollback-Recovery Protocols in Message-Passing systems*. ACM Computing Surveys, Volume 34, Number 3, September 2002, pp. 375-408.
3. J. Sherwani and N. Ali, N. Lotia, Z. Hayat and R. Buyya. *Libra: An Economy driven Job Scheduling System for Clusters*. Technical Report, The University of Melbourne, July 2002.
4. Leyuan Shi and Sigurdur Olafsson. *Hybrid Equipartitioning Job Scheduling Policies for Parallel Computer Systems*. Proceedings of the 37th IEEE Conference on Decision and Control , Tampa, Florida, December 1998, pp. 1704-1709.
5. Sivarama P. Dandamudi and Hai Yu. *Performance of Adaptive Space Sharing Processor Allocation Policies for Distributed-Memory Multicomputers*. Journal of Parallel Distributed Computing, Volume 58, 1999, pp. 109-125.
6. Thoai Nam, Tran Dinh Toan and Tran Vu Ngoc Tuong. *Resource Management and Scheduling on Supernode II*. School on Computational Sciences and Engineering: Theory and Applications, Hochiminh City, Vietnam, March 2005, pp. 41-52.
7. Thyagaraj Thanalapati and Sivarama Dandamudi. *An Efficient Adaptive Scheduling Scheme for Distributed Memory Multicomputers*. IEEE Transactions on Parallel and Distributed Systems, Volume 12, Issue 7, July 2001, pp. 758-768.
8. William Gropp, Ewing Lusk and Thomas Sterling. *Beowulf Cluster Computing with Linux*. The MIT Press, December 2003, ISBN:0262692929.
9. Yuet-Ning Chan, Sivarama P. Dandamudi and Shikharesh Majumdar. *Performance Comparison of Processor Scheduling Strategies in a Distributed-Memory Multicomputer System.* International Journal of Computers and Their Applications, Volume 9, Number 4, December 2002, pp. 227-240.
10. *CCS Homepage*. URL: http://www.uni-paderborn.de/pc2/projects/ccs.

11. *Condor Project Homepage.* URL: http://www.cs.wisc.edu/condor.
12. *PBS Hompage.* URL: http://www.openpbs.org.
13. *Maui Cluster Scheduler Homepage*. URL : http://www.clusterresources.com/products/maui.