Mục lục

[1. Giới thiệu tổng quan 2](#_Toc307653089)

[1.1. Giới thiệu điện toán đám mây 2](#_Toc307653090)

[1.2. Các bài toán mà điện toán đám mây giải quyết 4](#_Toc307653091)

[1.3. Lợi ích của điện toán đám mây 5](#_Toc307653092)

[2. Mục đích 6](#_Toc307653093)

[2.1. Ứng dụng vào điều kiện cụ thể của Việt Nam 6](#_Toc307653094)

[3. Nội dung 7](#_Toc307653095)

[3.1. Nghiên cứu điện toán đám mây 7](#_Toc307653096)

[3.1.1. Kiến trúc tổng quát của điện toán đám mây 7](#_Toc307653097)

[3.1.2. Các hình thức triển khai điện toán đám mây 10](#_Toc307653098)

[3.1.2.1. Đám mây công cộng 10](#_Toc307653099)

[3.1.2.2. Đám mây cá nhân 11](#_Toc307653100)

[3.1.2.3. Đám mây lai 12](#_Toc307653101)

[3.2.1.4. Đám mây cộng đồng 13](#_Toc307653102)

[3.1.3. Khác biệt giữa kiến trúc điện toán đám mây và kiến trúc client/server 14](#_Toc307653103)

[3.1.3.1. Khả năng phát triển hệ thống 14](#_Toc307653104)

[3.1.3.2. Những nhược điểm của điện toán đám mây 17](#_Toc307653105)

[3.1.4. Kiến trúc tổng quan về Windows Azure 18](#_Toc307653106)

[3.1.4.1. Dịch vụ tính toán 20](#_Toc307653107)

[3.1.4.2. Dịch vụ lưu trữ 22](#_Toc307653108)

[3.1.4.3. Fabric 23](#_Toc307653109)

[3.1.4.4. Các kịch bản sử dụng Windows Azure 26](#_Toc307653110)

[3.2. Ứng dụng trên điện toán đám mây 32](#_Toc307653111)

[3.3. Đề tài sổ liên lạc trực tuyến 33](#_Toc307653112)

[3.3.1. Giới thiệu tổng quan 33](#_Toc307653113)

[3.3.2. Lí do chọn đề tài sổ liên lạc trực tuyến 34](#_Toc307653114)

[3.3.3. Tính năng chính 35](#_Toc307653115)

[3.4. Phát triển bài toán 35](#_Toc307653116)

[3.4.1. Use Case Diagram 35](#_Toc307653117)

[3.4.2. Design 35](#_Toc307653118)

[3.4.2.1. Class Design Diagram 35](#_Toc307653119)

[3.4.2.2. Screen Design 35](#_Toc307653120)

[3.4.2.3. Database Design 35](#_Toc307653121)

[3.5. Kết quả thực hiện 35](#_Toc307653122)

[4. Kế hoạch 35](#_Toc307653123)

[5. Kết luận và hướng phát triển 35](#_Toc307653124)

# 1. Giới thiệu tổng quan

## 1.1. Giới thiệu điện toán đám mây

Ngày nay, đối với các công ty, doanh nghiệp, việc quản lý tốt và hiệu quả dữ liệu của riêng công ty cũng như dữ liệu khách hàng và đối tác là một trong những bài toán được ưu tiên hàng đầu và đang không ngừng gây khó khăn cho họ. Để có thể quản lý được nguồn dữ liệu đó, ban đầu các doanh nghiệp phải đầu tư, tính toán rất nhiều loại chi phí như chi phí cho phần cứng, phần mềm, mạng, chi phí cho quản trị viên, chi phí bảo trì, sửa chữa,… Ngoài ra họ còn phải tính toán khả năng mở rộng, nâng cấp thiết bị, phải kiểm soát việc bảo mật dữ liệu cũng như tính sẵn sàng cao của dữ liệu.

Từ một bài toán điển hình như vậy, chúng ta thấy được rằng nếu có một nơi tin cậy giúp các doanh nghiệp quản lý tốt nguồn dữ liệu đó, các doanh nghiệp sẽ không còn quan tâm đến cơ sở hạ tầng, công nghệ mà chỉ tập trung chính vào công việc kinh doanh của họ thì sẽ mang lại cho họ hiệu quả và lợi nhuận ngày càng cao hơn.

Thuật ngữ “điện toán đám mây (cloud computing)” ra đời bắt nguồn từ một trong những hoàn cảnh như vậy.

Thuật ngữ “điện toán đám mây” còn được bắt nguồn từ ý tưởng đưa tất cả mọi thứ như dữ liệu, phần mềm, tính toán, … lên trên mạng Internet. Chúng ta sẽ không còn trông thấy các máy PC, máy chủ của riêng các doanh nghiệp để lưu trữ dữ liệu, phần mềm nữa mà chỉ còn một số các “máy chủ ảo” tập trung ở trên mạng. Các “máy chủ ảo” sẽ cung cấp các dịch vụ giúp cho doanh nghiệp có thể quản lý dữ liệu dễ dàng hơn, họ sẽ chỉ trả chi phí cho lượng sử dụng dịch vụ của họ, mà không cần phải đầu tư nhiều vào cơ sở hạ tầng cũng như quan tâm nhiều đến công nghệ. Xu hướng này sẽ giúp nhiều cho các công ty, doanh nghiệp vừa và nhỏ mà không có cơ sở hạ tầng mạng, máy chủ để lưu trữ, quản lý dữ liệu tốt. Theo dự báo của Công ty Dữ liệu quốc tế IDC, thị trường các dịch vụ đám mây trên toàn thế giới sẽ đạt quy mô khoảng 43tỉ USD vào năm 2012. IDC cũng cho rằng những ứng dụng dịch vụ đám mây sẽ đạt tỷ lệ tăng trưởng hàng năm khoảng 27%, cao gấp khoảng 5 lần mô hình sử dụng các dịch vụ CNTT truyền thống.

Vậy “điện toán đám mây” là gì ? Nó có thể giải quyết bài toán trên như thế nào và có những đặc điểm nổi bật gì ?

Chúng ta sẽ tìm hiểu qua một vài định nghĩa được các chuyên gia trên thế giới đưa ra về “cloud computing”:

Theo Wikipedia:

*“Điện toán đám mây là một mô hình điện toán có khả năng co giãn linh động và các tài nguyên thường được ảo hóa để cung cấp như một dịch vụ trên mạng Internet”*.

Theo Ian Foster:

*“Điện toán đám mây là một mô hình điện toán phân tán có tính co giãn lớn hướng về mặt kinh tế, là nơi chứa các sức mạnh tính toán, kho lưu trữ, các nền tảng và các dịch vụ được trực quan, ảo hóa và co giãn linh động, sẽ được phân phối theo nhu cầu cho các khách hàng bên ngoài thông qua Internet”*.

*“Điện toán đám mây là một dạng thức điện toán cung cấp các tài nguyên ảo hóa và có quy mô dưới dạng dịch vụ qua mạng Internet. Người dùng không cần tới những kiến thức chuyên môn để quản lý hạ tầng công nghệ này bởi phần việc đó là dành cho các nhà cung cấp dịch vụ”*.

*“Điện toán đám mây là sự kết hợp giữa các khái niệm: Hạ tầng hướng dịch vụ (IaaS), Nền tảng hướng dịch vụ (PaaS), Phần mềm hướng dịch vụ (SaaS) và một số khái niệm công nghệ mới. Dịch vụ điện toán đám mây thường cung cấp các trực tuyến ứng dụng doanh nghiệp thông dụng, có thể truy xuất qua trình duyệt web trong khi phần mềm và dữ liệu được lưu trữ trên máy chủ của nhà cung cấp”.*



Hình 1 - Mọi thứ đều tập trung vào đám mây

## 1.2. Các bài toán mà điện toán đám mây giải quyết

- **Vấn đề về lưu trữ dữ liệu**: Dữ liệu được lưu trữ tập trung ở các kho dữ liệu khổng lồ. Các công ty lớn như Microsoft, Google có hàng chục kho dữ liệu trung tâm nằm rải rác khắp nơi trên thế giới. Các công ty lớn này sẽ cung cấp các dịch vụ cho phép doanh nghiệp có thể lưu trữ và quản lý dữ liệu của họ trên các kho lưu trữ trung tâm.

**- Vấn đề về sức mạnh tính toán**: Có hai giải pháp chính:

* Sử dụng các siêu máy tính (super-computer) để xử lý tính toán.
* Sử dụng các hệ thống tính toán song song, phân tán.

**- Vấn đề về cung cấp tài nguyên, phần mềm**:

Cung cấp các dịch vụ như IaaS (infrastructure as a service), PaaS (platform as a service), SaaS (software as a service).

Ở các trường đại học của Mỹ và một số quốc gia trên thế giới, điện toán đám mây được ứng dụng vào giáo dục ở rất nhiều khía cạnh. Nếu bạn là người có thẩm quyền quyết định trong một trường học, khi đứng trước một yêu cầu nâng cấp hệ thống cở sở hạ tầng dịch vụ để cải thiện chất lượng trong công tác dạy học cũng như để sinh viên tiếp cận được với những công nghệ mới, điều đầu tiên bạn nghĩ đến chính là thay mới toàn bộ hệ thống cơ sở vật chất hiện tại (chi phí rất lớn) nhưng điện toán đám mây cũng cấp cho bạn một giải pháp rất tối ưu mà bạn không cần phải bỏ ra quá nhiều chi phí bạn đầu. Khi bạn kết nối với dịch vụ của một nhà cung cấp điện toán đám mây, bạn sẽ tiếp cận được với những tài nguyên hiện đại nhất, đáp ứng đầy đủ cho yêu cầu giảng dạy và học tập của học sinh trong trường. Không những vậy việc áp dụng điện toán đám mây còn giúp cải thiện chất lượng học tập của học sinh tại nhà. Tất cả những bài tập làm của học sinh khi được lưu trữ trên đám mây có thể truy xuất được bất cứ ở đâu.

****

Hình 2 - Minh họa về các dịch vụ

## 1.3. Lợi ích của điện toán đám mây

**- Tiết kiệm**: nhanh chóng cải thiện với người dùng có khả năng cung cấp sẵn các tài nguyên cơ sở hạ tầng công nghệ một cách nhanh chóng và ít tốn kém.

**- Giảm chi phí**: chi phí được giảm đáng kể và chi phí vốn đầu tư được chuyển sang hoạt động chi tiêu. Điều này làm giảm rào cản cho việc tiếp nhận, chẳng hạn như cơ sở hạ tầng được cung cấp bởi đối tác thứ ba và không cần phải mua để dùng cho các tác vụ tính toán thực hiện một lần hay chuyên sâu mà không thường xuyên. Việc định giá dựa trên cơ sở tính toán theo nhu cầu thì tốt đối với những tùy chọn dựa trên việc sử dụng và các kỹ năng IT được đòi hỏi tối thiểu (hay không được đòi hỏi) cho việc thực thi.

**- Đa phương tiện**: sự độc lập giữa thiết bị và vị trí làm cho người dùng có thể truy cập hệ thống bằng cách sử dụng trình duyệt web mà không quan tâm đến vị trí của họ hay thiết bị nào mà họ đang dùng, ví dụ như PC, mobile. Vì cơ sở hạ tầng off-site (tức là được cung cấp bởi đối tác thứ ba) và được truy cập thông qua Internet, do đó người dùng có thể kết nối từ bất kỳ nơi nào.

**- Chia sẻ:** việc cho thuê nhiều để có thể chia sẻ tài nguyên và chi phí giữa một phạm vi lớn người dùng, cho phép:

* Tập trung hóa cơ sở hạ tầng trong các lĩnh vực với chi phí thấp hơn (chẳng hạn như bất động sản, điện, v.v.)
* Khả năng chịu tải nâng cao (người dùng không cần kỹ sư cho các mức tải cao nhất có thể).
* Cải thiện việc sử dụng và hiệu quả cho các hệ thống mà thường chỉ 10-20% được sử dụng.

**- Độ tin cậy**: độ tin cậy cải thiện thông qua việc sử dụng các site có nhiều dư thừa, làm nó thích hợp cho tính liên tục trong kinh doanh và khôi phục thất bại. Tuy nhiên, phần lớn các dịch vụ của điện toán đám mây có những lúc thiếu hụt và người giám đốc kinh doanh, IT phải làm cho nó ít đi.

**- Tính co giãn linh động:** tính co giãn linh động (theo nhu cầu) cung cấp tài nguyên trên một cơ sở mịn, tự bản thân dịch vụ và gần thời gian thực, không cần người dùng phải có kỹ sư cho chịu tải.

**- Hiệu suất**: hiệu suất hoạt động được quan sát và các kiến trúc nhất quán, kết nối lỏng lẽo được cấu trúc dùng web service như giao tiếp hệ thống.

**- Bảo mật:** Việc bảo mật cải thiện nhờ vào tập trung hóa dữ liệu, các tài nguyên chú trọng bảo mật, v.v… nhưng cũng nâng cao mối quan tâm về việc mất quyền điều khiển dữ liệu nhạy cảm. Bảo mật thường thì tốt hay tốt hơn các hệ thống truyền thống, một phần bởi các nhà cung cấp có thể dành nhiều nguồn lực cho việc giải quyết các vấn đề bảo mật mà nhiều khách hàng không có đủ chi phí để thực hiện. Các nhà cung cấp sẽ ghi nhớ (log) các truy cập, nhưng việc truy cập vào chính bản thân các audit log có thể khó khăn hay không thể.

**- Khả năng chịu đựng**: Khả năng chịu đựng xảy ra thông qua việc tận dụng tài nguyên đã được cải thiện, các hệ thống hiệu quả hơn. Tuy nhiên, các máy tính và cơ sở hạ tầng kết hợp là những thứ tiêu thụ năng lượng chủ yếu.

# 2. Mục đích

## 2.1. Ứng dụng vào điều kiện cụ thể của Việt Nam

Với xu hướng đường truyền Internet ngày càng trở nên nhanh và rẻ, việc tập trung dữ liệu và công việc xử lý về một trung tâm sẽ đem lại nhiều lợi ích như: dễ quản lý, giảm chi phí, tăng cường khả năng khai thác dữ liệu... Theo công ty nghiên cứu thị trường Gartner, gần 60% giám đốc CNTT ở châu Âu đang sử dụng các dịch vụ điện toán đám mây dù đa số họ vẫn chưa hiểu rõ hết về mô hình này. Gartner dự đoán, doanh thu của các dịch vụ điện toán đám mây trên toàn cầu, từ ngưỡng 56 tỷ đô-la Mỹ trong năm 2009 có thể đạt đến 150 tỷ đô-la vào năm 2013, tăng gần gấp ba lần. Song song đó, các ứng dụng dựa trên đám mây sẽ tiếp tục phát triển, như GooApps, Zoho...

Ở Việt Nam, điện toán đám mây đã bắt đầu có những tín hiệu lạc quan qua những cái bắt tay của các công ty công nghệ hàng đầu trong nước với các đối tác lớn như Microsoft, IBM, Trend Micro. Hồi cuối tháng Năm năm nay, tập đoàn FPT đã ký kết với Microsoft và Trend Micro để hợp tác, phát triển các dịch vụ điện toán đám mây về lưu trữ dữ liệu, dịch vụ hạ tầng. Trong khi đó, Công ty Phát triển công viên phần mềm Quang Trung và IBM Việt Nam cũng đã hợp tác xây dựng nền tảng đám mây để phục vụ nhu cầu của doanh nghiệp lẫn chính quyền; công ty phần mềm Misa dự kiến sẽ triển khai ứng dụng quản lý quan hệ khách hàng, quản lý nhân sự dựa trên nền tảng điện toán đám mây đến khách hàng vào cuối năm nay.

# 3. Nội dung

## 3.1. Nghiên cứu điện toán đám mây

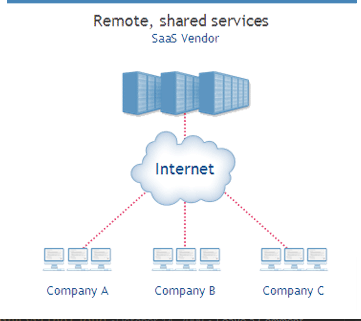
### 3.1.1. Kiến trúc tổng quát của điện toán đám mây



**Hình 3 – Kiến trúc tổng quát của đám mây**

Hình trên cho thấy, những dịch vụ trên đám mây được phân thành ba nhóm chính, bao gồm:

* ***Software as a service (SaaS)****:* là một mô hình triển khai ứng dụng mà ở đó người cung cấp cho phép người dụng sử dụng dịch vụ theo yêu cầu. Những nhà cung cấp SaaS có thể lưu trữ ứng dụng trên máy chủ của họ hoặc tải ứng dụng xuống thiết bị khách hàng, vô hiệu hóa nó sau khi kết thúc thời hạn. Các chức năng theo yêu cầu có thể được kiểm soát bên trong để chia sẻ bản quyền của một nhà cung cấp ứng dụng thứ ba.



**Hình 4 – Software as Service**

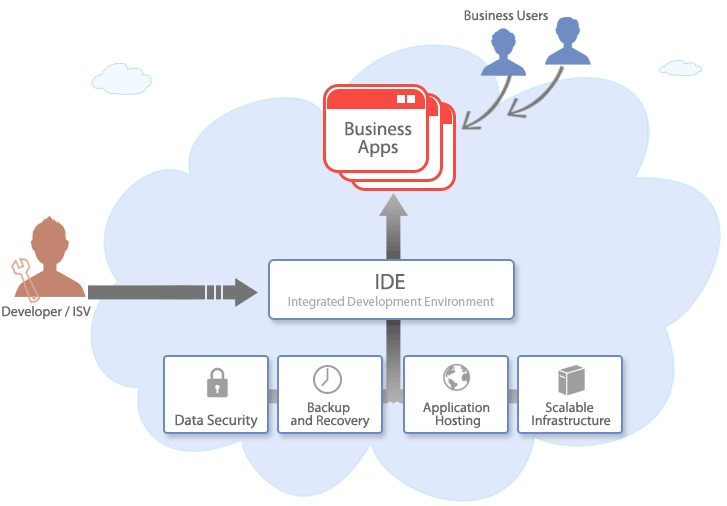
Một số ứng dụng sử dụng dịch vụ này:

* CRM
* Video Conferencing
* IT service management
* Kế toán
* Web analytics
* Web content management

Lợi ích lớn nhất của dịch vụ này mang lại là chi phí thấp. Nhà cung cấp dịch vụ có thể đưa ra các ứng dụng rẻ hơn và đáng tin hơn. Ngoài ra còn một số lợi ích khác như:

* Quen thuộc với môi trường World Wide Web
* Sử dụng ít nhân viên
* Sự tùy chỉnh: những ứng dụng trước đây rất khó tùy chỉnh và đòi hỏi hải cập nhật các bản vá lỗi. Ứng dụng SaaS dễ dàng tùy chỉnh và có thể đáp ứng chính xác yêu cầu của tổ chức
* Bảo mật: SSL (Secure Sockets Layer) được sử dụng rộng rãi và tin cậy
* ***Platform as a service (PaaS****):* hỗ trợ việc triển khai ứng dụng mà không quan tâm đến chi phí hay sự phức tạp của việc trang bị và quản lý các lớp phần cứng và phần mềm bên dưới, cung cấp tất cả các tính năng cần thiết để hỗ trợ chu trình sống đầy đủ của việc xây dựng và cung cấp một ứng dụng và dịch vụ web sẵn sàng trên internet mà không cần bất kì thao tác tải hay cài đặt phần mềm cho những người phát triển, quản lý tin học hay người dùng cuối.

Khi PaaS có sẵn như một dịch vụ, các developer có thể kiểm soát toàn bộ việc phát triển và triển khai ứng dụng. PaaS cho phép các developer tạo ra các ứng dụng web tùy chỉnh và phát hành nó một cách nhanh chóng, khi nhiều rắc rối như việc thiết lập hosting, servers, databases, quá trình tương tác người dùng và những frameworks được đóng gói.



**Hình 5 – Platform as Service**

Một số đặc trưng của Paas bao gồm:

* Phục vụ cho việc phát triển, kiểm thử, triển khai và vận hành ứng dụng giống như là môi trường phát triển tích hợp
* Cung cấp các công cụ khởi tạo với giao diện trên nền web
* Có kiến trúc đồng nhất
* Tích hợp dịch vụ web và cơ sở dữ liệu
* Hỗ trợ cộng tác nhóm phát triển
* Cung cấp các công cụ hỗ trợ tiện tích khác

Việc sử dụng dịch vụ này mang lại một số lợi ích:

* Ưu điểm trong những dự án tập hợp những công việc nhóm có sự phân tán về địa lý.
* Khả năng tích hợp nhiều nguồn của dịch vụ web.
* Giảm chi phí ngoài lề khi tích hợp các dịch vụ về bảo mật, khả năng mở rộng, kiểm soát lỗi…
* Giảm chi phí khi trừu tượng hóa công việc lập trình ở mức cao để tạo dịch vụ, giao diện người dùng và các yếu tố ứng dụng khác.
* Hướng việc sử dụng công nghệ để đạt được mục đích tạo điều kiện dễ dàng hơn cho việc phát triển ứng dụng đa người dùng cho những người không chỉ trong nhóm lập trình mà có thể kết hợp nhiều nhóm cùng làm việc.
* ***Infrastructure as a service (Iaas****):* cung cấp cơ sở hạ tầng máy tính (thường là môi trường ảo) như là một dịch vụ. Thay vì phải mua server, phần mềm, data-center hay thiết bị mạng, khách hàng có thể mua các tài nguyên như là một dịch vụ bên ngoài.

Các lợi ích mà dịch vụ này mang lại:

* Cung cấp tài nguyên như là dịch vụ: bao gồm cả máy chủ, thiết bị mạng, bộ nhớ, CPU, không gian đĩa cứng, trang thiết bị trung tâm dữ liệu.
* Khả năng mở rộng linh hoạt.
* Chi phí thay đổi tùy theo thực tế.
* Nhiều người thuê có thể cùng dùng chung trên một tài nguyên.
* Cấp độ doanh nghiệp: đem lại lợi ích cho công ty bởi một nguồn tài nguyên tích toán tổng hợp.

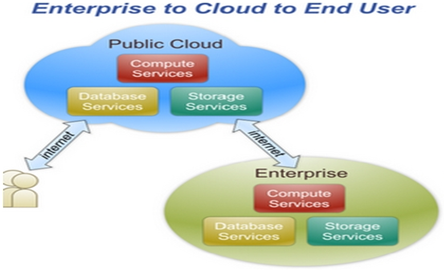
### 3.1.2. Các hình thức triển khai điện toán đám mây

#### 3.1.2.1. Đám mây công cộng

Là các dịch vụ điện toán đám mây được một bên thứ ba (người bán) cung cấp. Chúng tồn tại ngoài tường lửa công ty và được lưu trữ đầy đủ và được nhà cung cấp đám mây quản lý.

Các đám mây công cộng cố gắng cung cấp cho người dùng với các phần tử công nghệ thông tin tốt nhất. Cho dù đó là phần mềm, cơ sở hạ tầng ứng dụng hoặc cơ sở hạ tầng vật lý, nhà cung cấp đám mây chịu trách nhiệm về cài đặt, quản lý, cung cấp và bảo trì. Khách hàng chỉ chịu phí cho các tài nguyên nào mà họ sử dụng, vì thế cái chưa sử dụng được loại bỏ.

Tất nhiên điều này liên quan đến chi phí. Các dịch vụ này thường được cung cấp với quy ước về cấu hình, nghĩa là chúng được phân phối với ý tưởng cung cấp các trường hợp sử dụng phổ biến nhất. Các tùy chọn cấu hình thường là một tập hợp con nhỏ hơn so với những gì mà chúng đã có nếu nguồn tài nguyên đã được người tiêu dùng kiểm soát trực tiếp. Một điều khác cần lưu ý là kể từ khi người tiêu dùng có quyền kiểm soát một chút trên cơ sở hạ tầng, các quy trình đòi hỏi an ninh chặt chẽ và tuân thủ quy định dưới luật không phải lúc nào cũng thích hợp cho các đám mây chung.

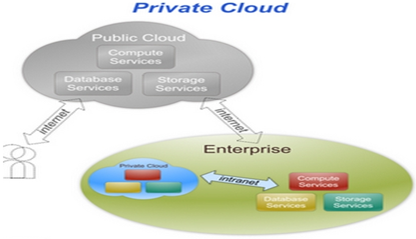


**Hình 6 – Đám mây công cộng**

#### 3.1.2.2. Đám mây cá nhân

Là các dịch vụ đám mây được cung cấp trong doanh nghiệp. Những đám mây này tồn tại bên trong tường lửa công ty và chúng được doanh nghiệp quản lý. Các đám mây riêng đưa ra nhiều lợi ích giống như các đám mây chung thực hiện với sự khác biệt chính: doanh nghiệp có trách nhiệm thiết lập và bảo trì đám mây này.

Sự khó khăn và chi phí của việc thiết lập một đám mây bên trong đôi khi có thể có chiều hướng ngăn cản việc sử dụng và chi phí hoạt động liên tục của đám mây có thể vượt quá chi phí của việc sử dụng một đám mây chung.Các đám mâycá nhân đưa ra nhiều lợi thế hơn so với loại cộng cộng. Việc kiểm soát chi tiết hơn trên các tài nguyên khác nhau đang tạo thành một đám mây mang lại cho công ty tất cả các tùy chọn cấu hình có sẵn. Ngoài ra, các đám mây riêng là lý tưởng khi các kiểu công việc đang được thực hiện không thiết thực cho một đám mây công cộng, do đúng với các mối quan tâm về an ninh và về quản lý.



**Hình 7 – Đám mây cá nhân**

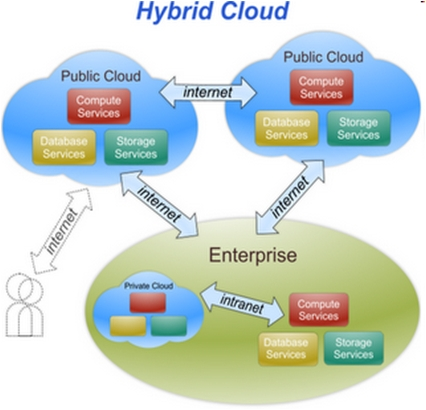
#### 3.1.2.3. Đám mây lai

Là một sự kết hợp của các đám mây công cộng và cá nhân. Những đám mây này thường do doanh nghiệp tạo ra và các trách nhiệm quản lý sẽ được phân chia giữa doanh nghiệp và nhà cung cấp đám mây công cộng. Đám mây lai sử dụng các dịch vụ có trong cả không gian công cộng và cá nhân.

Các đám mây lai là câu trả lời khi một công ty cần sử dụng các dịch vụ của cả hai đám mây riêng và công cộng. Theo hướng này, một công ty có thể phác thảo các mục tiêu và nhu cầu của các dịch vụ và nhận được chúng từ đám mây công cộng hay cá nhân, khi thích hợp.

Một đám mây lai được xây dựng tốt có thể phục vụ các quy trình nhiệm vụ-tới hạn, an toàn, như nhận các khoản thanh toán của khách hàng, cũng như những thứ là không quan trọng bằng kinh doanh, như xử lý bảng lương nhân viên.

Hạn chế chính với đám mây này là sự khó khăn trong việc tạo ra và quản lý có hiệu quả một giải pháp như vậy. Phải có thể nhận được và cung cấp các dịch vụ lấy từ các nguồn khác nhau như thể chúng có nguồn gốc từ một chỗ và tương tác giữa các thành phần riêng và chung có thể làm cho việc thực hiện thậm chí phức tạp hơn nhiều. Do đây là một khái niệm kiến trúc tương đối mới trong điện toán đám mây, nên cách thực hành và các công cụ tốt nhất về loại này tiếp tục nổi lên và bất đắc dĩ chấp nhận mô hình này cho đến khi hiểu rõ hơn.

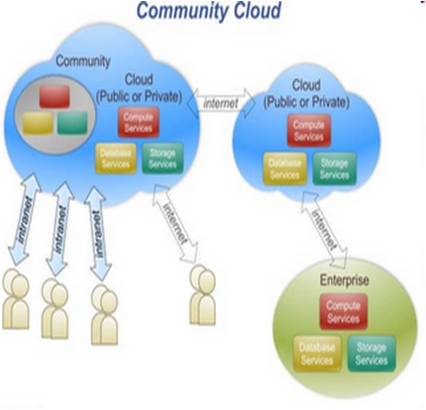


**Hình 8 – Đám mây lai**

#### 3.2.1.4. Đám mây cộng đồng

Là các đám mây được chia sẻ bởi một số tổ chức và hỗ trợ một cộng đồng cụ thể có mối quan tâm chung (ví dụ như chung sứ mệnh, yêu cầu an ninh, chính sách,.. ). Nó có thể được quản lý bởi các tổ chức hoặc một bên thứ ba.

Một đám mây cộng đồng có thể được thiết lập bởi một số tổ chức có yêu cầu tương tự và tìm cách chia sẻ cơ sở hạ tầng để thực hiện một số lợi ích của điện toán đám mây. Tùy chọn này là tốn kém hơn nhưng có thể đáp ứng về sự riêng tư, an ninh hoặc tuân thủ các chính sách tốt hơn.



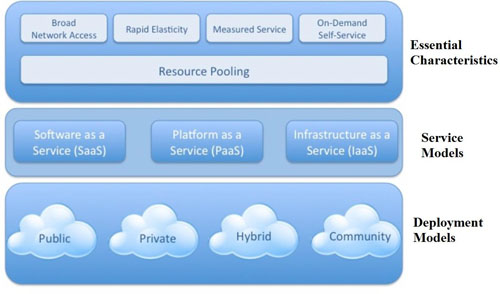
**Hình 9 – Đám mây cộng đồng**

### 3.1.3. Khác biệt giữa kiến trúc điện toán đám mây và kiến trúc client/server

#### 3.1.3.1. Khả năng phát triển hệ thống

Điện toán đám mây "mở" hơn, quan trọng hơn, giá rẻ hơn,. . . Như vậy rất tiện lợi cho các doanh nghiệp và tổ chức hoạt động vì giảm chi phí. Để trả lời ngắn gọn câu hỏi: “Điện toán đám mây có thật sự hữu ích cho một tổ chức, công ty hay không?” chúng ta phải xem xét: quy mô, hình thức hoạt động,…Đặc biệt nếu dữ liệu là vô cùng quan trọng đối với họ, chắc hắn người ta sẽ xây dựng một hệ thống lưu trữ riêng và nó được vận hành như hệ thống lưu trữ của các ngân hàng. Đối với các doanh nghiệp nhỏ mà dữ liệu không quan trọng lắm, điện toán đám mây sẽ là giải pháp tối ưu do họ không phải chi phí đầu tư, quản lý, bảo trì, vận hành hệ thống lưu trữ.

Đặc tính của dịch vụ điện toán đám mây (Cloud Computing Service): điện toán đám mây có năm tính chất nổi bật so với mô hình truyền thống:

****

* **Khả năng co giãn (Rapid elasticity)**

Một đặc tính nổi bật của điện toán đám mây là khả năng tự động mở rộng hoặc thu nhỏ hệ thống theo yêu cầu người dùng (hệ thống sẽ tự mở rộng hoặc thu hẹp bằng cách thêm hoặc giảm bớt tài nguyên).

Một người dùng ký hợp đồng thuê một Server gồm 4 CPU. Nếu lượng truy cập thấp chỉ cần 1 CPU là đủ, khi đó hệ thống quản lý của nhà cung cấp dịch vụ sẽ tự ngắt bớt 3 CPU, người dùng không phải trả phí cho 3 CPU nói trên và chúng được đưa sang phục vụ người dùng khác. Đến khi nhu cầu tăng tức là lượng truy cập tăng, hệ thống ngay lạp tức sẽ tự động thêm CPU vào, nếu nhu cầu vượt quá 4 CPU thì người dùng trả phí theo hợp đồng đã ký với nhà cung cấp dịch vụ điện toán đám mây.

Khả năng co giãn nhanh và linh hoạt giúp cho nhà cung cấp dịch vụ điện toán đám mây tận dụng tài nguyên dư thừa phục vụ được nhiều khách hàng, người dùng giảm chi phí vì họ chỉ phải trả tiền cho những tài nguyên thực sự dùng.

* **Dịch vụ theo nhu cầu (On-demand self-service)**

Người dùng gửi yêu cầu thông qua trang web cung cấp dịch vụ, hệ thống của nhà cung cấp sẽ đáp ứng để người dùng có thể tự phục vụ như: tăng – giảm thời gian sử dụng server và dung lượng lưu trữ, … mà không cần phải trực tiếp yêu cầu nhà cung cấp dịch vụ, tức là mọi nhu cầu khách hàng đều được xử lý trên internet.

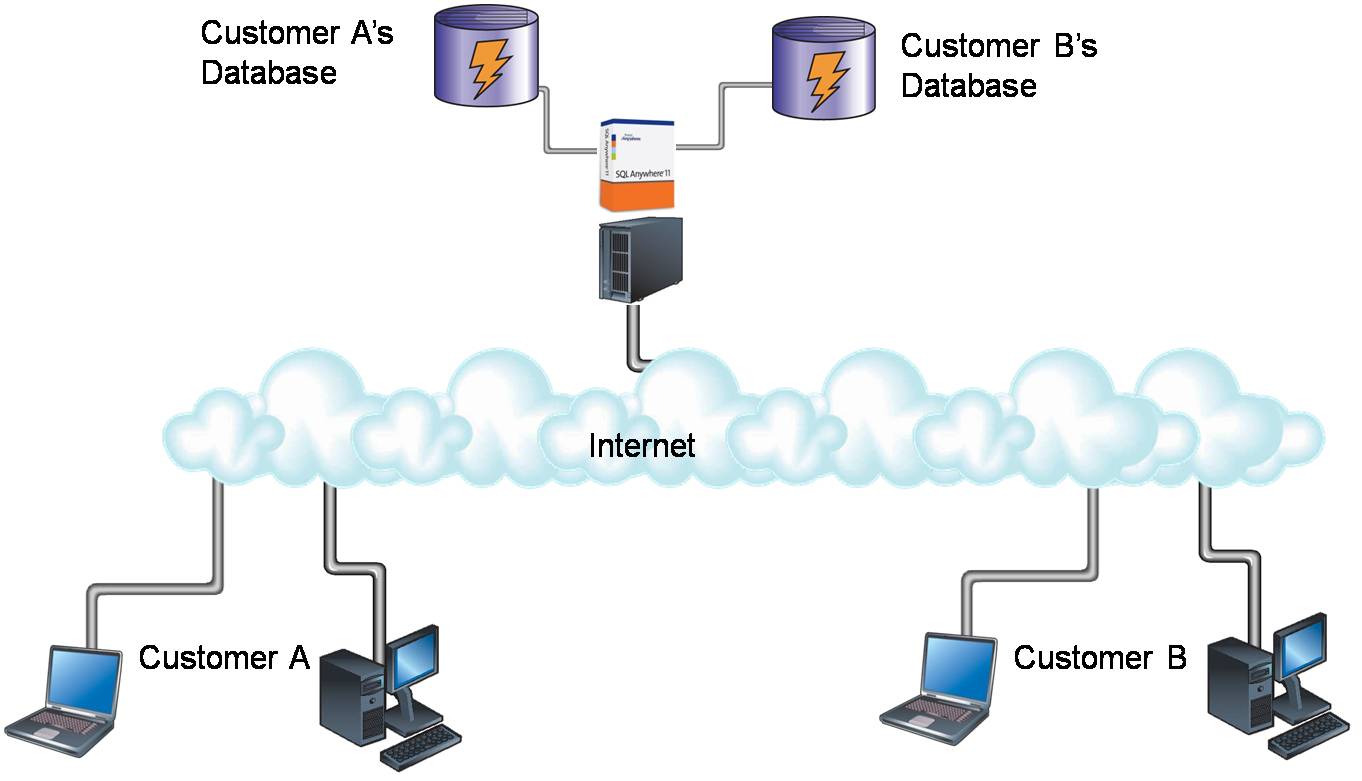
* **Truy xuất diện rộng**

Cloud Computing Service là tập hợp các dịch vụ công nghệ thông tin được cung cấp thông qua môi trường internet, ở đó người dùng thích dịch vụ gì thì dùng dịch vụ ấy, dùng bao nhiêu trả bấy nhiêu, được lựa chọn những dịch vụ tốt nhất ở bất cứ đâu ào bất cứ lúc nào. Như vậy người dùng có kết nối internet là có thể sử dụng dịch vụ, Cloud Computing Service không yêu cầu người dùng phải có khả năng xử lý cao, người dùng có thể truy xuất bằng các thiết bị di dộng như điện thoại, PDA, laptop.

* **Dùng chung tài nguyên và điều tiết dịch vụ:**

Nhà cung cấp dịch vụ cho phép người dùng dùng chung tài nguyên do họ cung cấp dựa trên mô hình “multi-tenant”, tài nguyên được phân phát rất linh hoạt tùy theo nhu cầu của người dùng. Khi nhu cầu của một người dùng nào đó giảm xuống, lập tức phần tài nguyên dư thừa sẽ được phục vụ cho người dùng khác. Nếu một người dùng 4 CPU từ 7 - đến 11 giờ hàng ngày, một người dùng khác thuê 4 CPU tương tự 13 giờ đến 17 giờ hàng ngày thì họ có thể dùng chung 4 CPU đó.

Cloud Computing Service dựa trên công nghệ ảo hóa, tài nguyên ở đây đa phần là tài nguyên ảo, chúng được cấp phát linh hoạt tùy theo nhu cầu (động) của từng người dùng khác nhau, nhà cung cấp dịch vụ có thể phục vụ nhiều người dùng hơn so với cách cấp phát tài nguyên (tĩnh) truyền thống.  
Hệ thống Cloud Computing Service tự động kiểm soát và tối ưu hóa sử dụng tài nguyên bao gồm: dung lượng lưu trữ, đơn vị xử lý, băng thông,.. Lượng tài nguyên sử dụng có thể được theo dõi, kiểm soát và báo cáo một cách minh bạch cho cả hai phía nhà cung cấp dịch vụ và người dùng.



#### 3.1.3.2. Những nhược điểm của điện toán đám mây

* **Data lock-in**

Ngày nay, khả năng tương tác giữa các nền tảng khác nhau của các phần mềm đã được cải thiện, nhưng các hàm API (Application Programming Interface) của Cloud Computing vẫn chưa được chuẩn hóa nên nếu một người dùng viết một ứng dụng trên nền tảng của nhà cung cấp dịch vụ thì ứng dụng đó không thể chạy được trên nền tảng của nhà cung cấp dịch vụ khác. Như vậy người dùng phụ thuộc nhà cung cấp dịch vụ là điều bất lợi.

Nếu các các nhà cung cấp Cloud Computing Service cùng nhau chuẩn hóa API, người dùng có thể phát triển ứng dụng trên nền tảng của nhiều nhà cung cấp dịch vụ. Nếu hệ thống cung cấp dịch vụ nào đó gặp sự cố thì dữ liệu người dùng không mất vì nó đã nằm đâu đó trên hệ thống của các nhà cung cấp dịch vụ khác. Nếu như cách này được thực hiện, chắc chắn sẽ dẫn đến cuộc cạnh tranh về giá, đến đây khi lựa chọn dịch vụ người ta phải cân nhắc:

* Thứ nhất, chất lượng dịch vụ tương xứng với giá mà người sử dụng trả cho nhà cung cấp dịch vụ, dù đắt người ta vẫn dùng, tức là “đắt xắt ra miếng”.
* Thứ hai, giảm data lock – in và chuẩn hóa các API sẽ dẫn đến khả năng: “cơ sở hạ tầng, phần mềm có thể chạy trên private cloud hoặc public cloud”.

Khi người dùng lưu trữ dữ liệu trên hệ thống của nhà cung cấp dịch vụ, câu hỏi đặt ra là: có gì đảm bảo dữ liệu an toàn, không rò rỉ? Về mặt công nghệ, hiện nay vẫn chưa có cách nào hiệu quả để giải quyết vấn đề trên, do đó người dùng thường chỉ lựa chọn những nhà cung cấp lơn và có uy tín.

* **Bảo mật và kiểm tra dữ liệu**

Như đã nói ở trên, dữ liệu lưu trên đám mây có an toàn không? Nhưng chắc chắn xác suất bị người khác khác truy xuất rất cao, đây thực sự là một thách thức trong bảo mật dữ liệu. Trước hết người dùng phải mã hóa dữ liệu trước khi đưa lên cloud để lưu trữ, khi sử dụng tất nhiên phải giải mã trên PC của họ. Người dùng ghi nhận thông tin hệ thống đã sử lý cùng với sử dụng các hệ điều hành ảo khi cung cấp dịch vụ IaaS sẽ làm cho ứng dụng của mình khó bị tấn công hơn.

Ngoài ra, bảo mật dữ liệu phụ thuộc con người, luật bảo vệ người dùng cloud computing service, nghĩa là nhà cung cấp dịch vụ phải cho người dùng tùy ý lựa chọn vị trí lưu trữ và chịu trách nhiệm pháp lý bảo đảm dữ liệu của người dùng không bị rò rỉ, ngược lại phải bồi thường theo luật ra bên ngoài.

* **Tắc nghẽn trên đường truyền dữ liệu và hiệu quả PC**

Có những ứng dụng khi bắt đầu chạy thì dữ liệu ít, càng về sau dữ liệu càng nhiều, có ứng dụng chạy trên Cloud và có thể lưu ở các vị trí khác nhau, khi chạy ứng dụng này phát sinh “vận chuyển dữ liệu giữa các data center”. Người dùng phải trả phí vận chuyển dữ liệu giữa các data center, ứng dụng chạy càng về sau thì chi phí này càng tăng lên, đây là điều phải cân nhắc.

Khi nhiều máy tính ảo cùng chạy, thì vấn đề chia sẽ về CPU hay bộ nhớ đạt hiệu quả cao, nhưng vấn đề giao tiếp IO của các máy ảo này gây ra nhiều vấn đề liên quan đến hiệu suất máy tính.

Để giảm ảnh hưởng của việc truy xuất vào ổ cứng. Ta có thể dùng flash để hạn chế trong giảm hiệu suất này.

* **Nhu cầu lưu trữ người dùng**

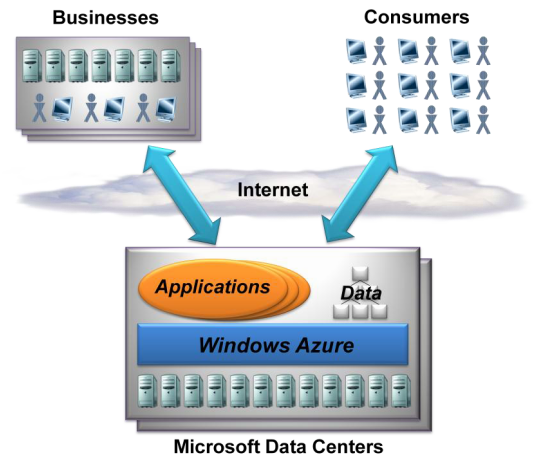
Mặc dù Cloud Computing đáp ứng linh hoạt nhu cầu lưu trữ của người nhưng lại gây khó khăn trong quản lý hệ thống lưu trữ, chẳng hạn một người sử dụng mua một khoảng dung lượng thì phải cung cấp cho người đó bao nhiêu là tối ưu, vừa đủ cho người dùng hay nhiều hơn yêu cầu, tăng độ phức tạp cấu trúc dữ liệu (cấu trúc dữ liệu làm sao hổ trợ vấn đề lưu trữ, vấn đề duyệt, vấn đề mở rộng...), hiệu suất truy xuất dữ liệu trong ổ cứng không cao (nếu phục vụ nhu cầu của người sử dụng thì hệ thống lưu trữ của mình có thể dễ bị hiện tượng phân mảnh trong lưu trữ).

Như vậy, làm sao tạo ra một hệ thống lưu trữ tiện lợi, đáp ứng vụ nhu cầu và khả năng lưu trữ của người sử dụng đang là vấn đề phức tạp phải giải quyết của các nhà cung cấp Cloud Computing Service.

### 3.1.4. Kiến trúc tổng quan về Windows Azure

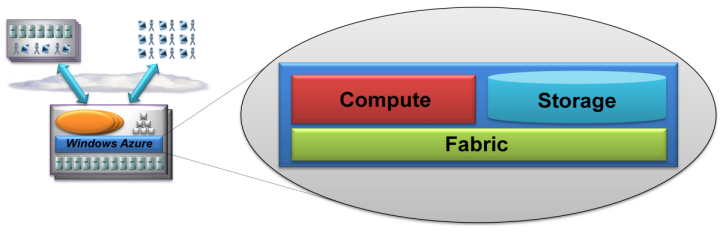
Với xu hướng phát triển của điện toán đám mây hiện này, dĩ nhiên là Microsoft không đứng sau trào lưu này. Microsoft đã cho ra đời Windows Azure Platform nhằm mang đến cho cộng đồng các nhà phát triển ứng dụng cơ hội được xây dựng và cung cấp các dịch vụ trực tuyến trên nền tảng cơ sở hạ tầng Windows.

Nhìn một cách tổng quan, Windows Azure là một hệ điều hành dùng để chạy các ứng dụng Windows và lưu dữ liệu của nó trên đám mây. Nhưng khác với một hệ điều hành bình thường, người dùng phải cài đặt và chạy trên máy tính của mình, Windows Azure là một dịch vụ: Khách hàng dùng nó để chạy ứng dụng, và lưu trữ dữ liệu trên các máy ở trung tâm dữ liệu Microsoft, có thể truy cập qua interner. Các ứng dụng này có thể cung cấp dịch vụ cho doanh nghiệp, khách hàng, hay cả hai.



Hình: Ứng dụng Windows Azure chạy trên trung tâm dữ liệu của Microsoft

Windows Azure gồm các thành phần cơ bản như sau:



Hình: Các thành phần của Windows Azure: Compute Service, Storage Service và Fabric

Giống như tên của nó, dịch vụ tính toán (Compute service) sẽ chạy ứng dụng trong khi dịch vụ Lưu trữ (Storage service) lưu dữ liệu. Thành phần thứ ba, Windows Azure Fabric, cung cấp cách thức thông dụng để quản lý và theo dõi các ứng dụng sử dụng nển tảng đám mây này.

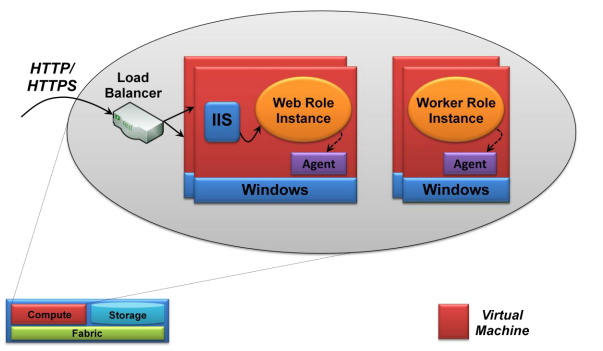
#### 3.1.4.1. Dịch vụ tính toán

Dịch vụ tính toán Windows Azure có thể chạy nhiều kiểu ứng dụng khác nhau. Mục tiêu chính của kiến trúc này, là hỗ trợ các ứng dụng có lượng người sử dụng truy cập đồng thời cực lớn. Có thể đạt được mục tiêu này bằng cách tăng cường sử dụng nhiều máy chủ lớn hơn.Nhưng thay vì như vậy, Windows Azure được thiết kế để hỗ trợ ứng dụng giảm xuống, chạy nhiều bản sao của cùng một mã nguồn trên nhiều máy chủ khác nhau.

Để đạt được điều này, ứng dụng Windows Azure có thể có nhiều thể hiện (instance), mỗi thể hiện (instance) được thực thi trên một máy ảo.

Để chạy một ứng dụng, lập trình viên truy cập Windows Azure portal thông qua trình duyệt, đăng nhập với một Windows Live ID. Sau đó, lập trình viên tạo ra một tài khoản *hosting* để chạy ứng dụng, hoặc một tài khoản lưu trữ (storage) để lưu trữ dữ liệu, hoặc cả hai. Một khi lập trình viên có tài khoản hosting thì có thể upload ứng dụng của mình, chỉ ra bao nhiêu thể hiện mà ứng dụng cần, cũng như cấu hình của máy ảo, Windows Azure sẽ tạo ra các máy ảo tương ứng để chạy ứng dụng. Lập trình viên, chỉ có thể thấy được trạng trái của ứng dụng được triển khai, thông qua Windows Azure portal. Một khi ứng dụng được triển khai, nó hoàn toàn được quản lý bởi Windows Azure. Điều duy nhất bạn phải làm là, chỉ ra các thông số sử dụng cho ứng dụng, còn lại, việc triển khai, tính mở rộng, availability, nâng cấp, chuản bị phần cứng server đều được thực hiện bởi Windows Azure cho các ứng dụng đám mây.

Compute service hỗ trợ 2 loại thể hiện, một loại gọi là Web role, và một loại gọi là Worker role.



Hình: Ứng dụng Windows Azure có thể chứa Web role hoặc Worker role, hoặc cả hai.

Một thể hiện Web role có thể chấp nhận một request HTTP/HTTPS. Để cho phép điều này, nó chạy trên một máy ảo có Internet Information Services (IIS) 7. Lập trình viên có thể tạo ra Web role bằng ASP.NET, WCF, hay bất kì kĩ thuật .NET nào có thể hoạt động được với IIS 7. Ngoài ra, lập trình viên có thể viết các ứng dụng với native code - việc sử dụng.NET Framework là không yêu cầu. Có nghĩa là có thể upload và chạy các ứng dụng sử dụng kĩ thuật khác, ví dụ PHP và Java. Khi một request được gửi đến Web role, nó sẽ được truyền qua bộ cân bằng tải đến các thể hiện của Web role trong cùng một ứng dụng. Do đó, không đảm bảo rằng, các yêu cầu từ một người dùng có thể được gởi đến cùng một thể hiện của ứng dụng.

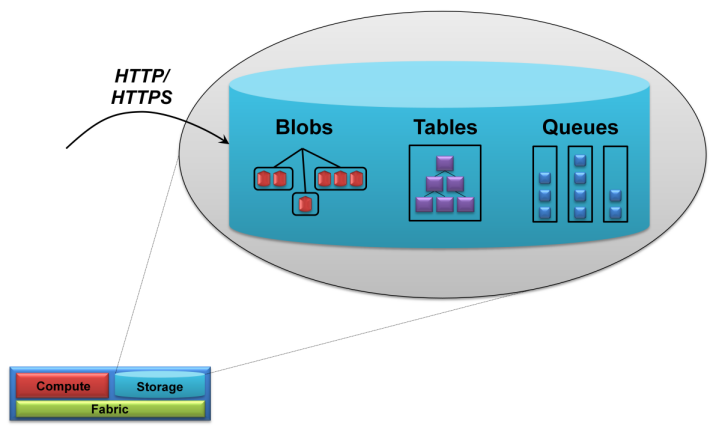
Một thể hiện Worker role không giống như Web role, nó không chấp nhận request từ bên ngoài, các máy ảo của nó không chạy IIS. Một Worker role cho bạn khả năng để chạy các xử lý ngầm liên tục trên đám mây. Một Worker role có thể làm việc với queue, table, blob trong dịch vụ lưu trữ. Nó chạy hoàn toàn độc lập với thể hiện Web role, mặc dù có thể cùng thuộc một phần của dịch vụ. Việc liên lạc giữa Web role và Worker role có thể thông qua queue của dịch vụ lưu trữ.

Lập trình viên có thể chỉ sử dụng thể hiện Web role, hay Worker role, hoặc kết hợp cả hai để tạo ra ứng dụng Windows Azure. Có thể sử dụng Windows Azure portal để thay đổi số lượng thể hiện của Web role, Worker role tùy theo yêu cầu của ứng dụng

Khi chạy các thể hiện Web role hay Worker role, các máy ảo cũng chạy đồng thời các tác nhân (Windows Azure *agent*). Các tác nhân để phục vụ cho sự tương tác hệ giữa các thể hiện với Windows Azure Fabric. Các agent này trình bày các API được định nghĩa để các thể hiện có thể làm một số việc như: ghi chép, tìm thư mục gốc của tài nguyên lưu trữ cục bộ trên máy ảo của nó.

#### 3.1.4.2. Dịch vụ lưu trữ

Dịch vụ lưu trữ trong Windows Azure hỗ trợ 3 kiểu dịch vụ: blob, table và queue.



Hình 4 Bộ lưu trữ Windows Azure gồm: Blob, Table và Queue

Những kiểu dịch vụ này hỗ trợ cục bộ cũng như truy cập trực tiếp thông qua REST API. Bảng so sánh giữa các kiểu lưu trữ này:

|  |  |  |  |
| --- | --- | --- | --- |
| Đặc trưng | Blob | Queue | Table |
| URL schema | http://[Storage  Account].blob.core.wi  ndows.net/[Container  Name]/[Blob Name] | http://[Storage  Account].queue.core.w  indows.net/[Queue  Name] | http://[Storage  Account].table.core.windows.  net/[Table  Name]?$filter=[Query] |
| Kích thước tối đa | 50GB | 8K(String) | Thiết kế cho Terabytes dữ liệu |
| Sử dụng | Lưu trữ dữ liệu binary lớn | Giao tiếp giữa các dịch vụ | Lưu trữ dữ liệu có cấu trúc kích thước nhỏ. |
| Tham khảo API | <http://msdn.microsoft.com/en-us/library/dd135733.aspx> | <http://msdn.microsoft.com/en-us/library/dd179363.aspx> | <http://msdn.microsoft.com/en-us/library/dd179423.aspx> |

Cách đơn giản nhất để lưu trữ dữ liệu trong Windows Azure storage là sử dụng Blob. Một blob chứa dữ liệu nhị phân. Blob cấu trúc lưu trữ đơn giản như sau: Mỗi tài khoản lưu trữ có một hoặc nhiều container, mỗi container chứa một hoặc nhiều blob. Kích thước Blob có thể lớn đến 50GB, chúng có thể chứa thêm metadata. Ví dụ: nơi chụp của tấm ảnh, hay ca sĩ thể hiện bài hát trong file MP3…

Bộ lưu trữ Windows Azure cũng cung cấp Table. Tuy nhiên, nó không phải là bảng quan hệ như trong SQL. Thực tế, dữ liệu lưu trữ bên trong nó là một hệ thống các thực thể với các thuộc tính. Hơn cả việc sử dụng SQL, một ứng dụng có thể truy cập dữ liệu của Table bằng ADO.NET data Service hoặc LINQ. Một bảng có thể sẽ rất lớn, với hàng tỉ thực thể chứa hàng terabyte dữ liệu. Và Bộ lưu trữ Windows Azure có thể phân vùng cho nó qua nhiều máy chủ khác nhau để tăng hiệu suất.

Cả blob và table đều phục vụ cho mục đích lưu trữ và truy xuất dữ liệu. Trong khi đó, queue nhằm phục vụ mục đích khác - đó là sự liên lạc giữa Web role và Worker role. Trong đó, Web role khi nhận được yêu cầu từ người dùng, sẽ ghi thông điệp vào một hàng đợi, mô tả công việc cho Worker role. Một Worker role chờ thông điệp này, lấy thông điệp, và thực hiện tác vụ yêu cầu.

Bộ lưu trữ Windows Azure có thể được truy cập từ một ứng dụng Windows Azure hoặc từ một ứng dụng khác. Trong cả 3 trường hợp, cả ba cách lưu trữ của dịch vụ lưu trữ Windows Azure đều có thể sử dụng REST để truy xuất dữ liệu. Mọi thứ đều được đặt tên qua URL và được truy xuất thông qua các thao tác HTTP chuẩn. Ngoài ra, còn hỗ trợ ADO.NET Data Service, nhưng chỉ có thể sử dụng cho các ứng dụng .NET, còn các ứng dụng khác ví dụ Java thì chỉ có thể sử dụng REST.

#### 3.1.4.2.1. Blob

Các thành phần của Windows Azure Blob.

* **Tài khoản lưu trữ** – Tất cả truy xuất đến bộ lưu trữ Windows Azure được thực hiện thông qua tài khoản lưu trữ.
  + Một tài khoản có thể có nhiều Container.

Hình 1 Khái niệm bộ lưu trữ Blob

* + **Blob Container** – Một container bao gồm nhiều blob.
  + Chính sách chia sẻ được thiết lập ở cấp độ container. Hiện tại hai thuộc tính "Public READ" và "Private" được hỗ trợ. Khi một container có thuộc tính là “Public READ”,thì tất cả nội dung của nó có thể được đọc bởi bất kì người dùng mà không yêu cầu chứng thực. Khi một container có thuộc tính là “Private”, thì chỉ có người sở hữu tương ứng mới có thể truy xuất blob trong container với truy xuất được chứng thực.
  + Một container có thể có metadata kết hợp với chúng. Metadata ở dạng cặp <tên, giá trị>, và chúng lên đến 8KB kích thước với mỗi container.
  + **Blob** – Blob được lưu trữ trong Container. Kích thước mỗi blob có thể lên đến 50GB. Một blob có một tên duy nhất trong một container. Blob có thể có metadata kết hợp với chúng, cặp <tên, giá trị>, và chúng lên đến 8KB kích thước cho mỗi blob.

Đường dẫn URI cho một blob được cấu trúc như sau:

http://<account>.**blob**.core.windows.net/<container>/<blobname>

Tất cả các giao tiếp đến Windows Azure Blob được thực hiện thông qua giao tiếp HTTP REST PUT/GET/DELETE.

Các lệnh HTTP/REST được hỗ trợ để thực thi các thao tác blob bao gồm :

* PUT Blob – Thêm một blob mới hoặc ghi đè blob đã tồn tại.
* GET Blob – Lấy một thực thể blob, hoặc lấy một dãy các byte trong blob sử dụng thao tác GET chuẩn HTTP.
* DELETE Blob – Xóa một blob đã tồn tại.
* Copy Blob – Sao chép một blob từ blob nguồn đến blob đích trong cùng tài khoản lưu trữ. Nó sẽ sao chép toàn bộ blob, bao gồm metadata, thuộc tính, và blocklist. Bạn có thể sử dụng CopyBlob cùng với DeleteBlob để đổi tên một blob, để di chuyển một blob giữa các containers, hoặc tạo bảng sao dự phòng cho blob đang tồn tại của bạn.
* Get Block List - Lấy danh sách các block đã được tải lên như một phần của blob. Có hai loại danh sách block:
  + Danh sách block đã commit – Đây là danh sách block đã được commit thành công qua PutBlockList vào một blob.
  + Danh sách block chưa commit – Đây là danh sách các block đã được tải lên cho một blob những chưa được commit vào blob.

Bạn có thể tải một blob có kích thước lên đến 64MB chỉ sử dụng một request PUT blob . Để tải lên các blob có kích thước lên đến 50GB, bạn cần phải ghép các block lại với nhau.

Một trong những chiến lược chính cho Windows Azure Blob là cho phép tải lên các blob có kích thước lên đến nhiều GB. Điều này được cung cấp bởi Windows Azure Blob qua những bước sau:

* Cắt Blob cần tải lên thành nhiều khối liên tiếp nhau. Vd, Một phim 10GB có thẻ cắt thành 2500 khối, mỗi khối có kích thước 4MB, khối đầu tiên đại diện cho byte thứ 1 đến 4194304, khối thứ 2 đại diện cho byte 4194305 đến 8388608.
* Đặt mỗi khối một Tên/ ID duy nhất. ID duy nhất này được giới hạn bởi tên blob được tải lên. Ví dụ, block đầu tiên được gọi là “Block 0001”, block thứ hai “Block 0002”, …
* PUT mỗi block vào đám mây.
* Sau khi tất cả các block đã được lưu trữ trong bộ lưu trữ Windows Azure, sau đó chúng ta commit danh sách các block đã tải lên vào blob mà chúng thuộc về. Điều này được thực hiện với một lênh PUT chỉ định URL ở trên với truy vấn xác định rằng đây là lệnh Blocklist. Sau đó, header HTTP chứa danh sách các block đã được commit cho khối này. Khi thao tác này thực hiện thành công, danh sách các block, theo thứ tự mà chúng được liệt kê, bây giờ đại diện cho phiên bản có thể đọc được của các blob. Blob sau đó có thể được đọc bằng cách sử dụng lệnh GET blob.

Hình dưới đây kết hợp các khối vào khái niệm dữ liệu Windows Azure Blob.

Hình 2 Khái niệm lưu trữ Blob – Thêm Block

Như mô tả trước đó, blob có thể được truy xuất qua PUT và GET bằng cách sử dụng URL sau:

http://<account>.**blob**.core.windows.net/<container>/<blobname>

Ví dụ hình 2, một lệnh PUT có thể được sử dụng để đặt các ảnh theo URL sau:

http://sally.**blob**.core.windows.net/pictures/IMG001.JPG

http://sally.**blob**.core.windows.net/pictures/IMG002.JPG

Khi sử dụng một lệnh PUT đơn, block có kích thước lên đến 64MB có thể được lưu trữ. Để lưu trữ các blob lớn hơn 64MB và lên đến 50GB, đầu tiên cần PUT tất cả các block, và sau đó PUT blocklist để hợp thành phiên bản có thể đọc được của blob này. Trong hình 2 ở trên,chỉ sau khi các block đã được put và commit thành blob, thì blob có thể được đọc theo URL sau:

http://sally.**blob**.core.windows.net/pictures/MOV1.AVI

**Trừu tượng hóa dữ liệu Block**

Mỗi block được xác định bởi Block ID và nó được giới hạn bởi tên blob. Vì thế những blob khác nhau có thể có các block với cùng ID. Mỗi block có kích thước lên đến 4MB, và các block trong cùng blob có thể có kích thước khác nhau. Windows Azure Blob cung cấp các thao tác cấp độ Block như sau:

* PUT block – tải một block vào một blob. Chú ý một block được tải lên thành công với thao tác PUT block không trở thành một phần của blob chỉ khi nào nó được commit vào blocklist với thao tác PUT blocklist.
* PUT blocklist – commit một blob bằng cách xác định danh sách các ID block tạo nên blob. Các block được xác định trong hoạt động này phải được tải lên thành công bằng cách sử dụng lời gọi PUT block. Thứ tự các block trong thao tác PUT blocklist, sẽ hợp thành phiên bản có thể đọc của blob.
* GET blocklist – lấy danh sách block đã được commit trước đó cho blob bằng thao tác PUT blocklist. Danh sách block được trả ra định rõ ID và kích thước của mỗi block. Chức năng này có thể được sử dụng để lấy các danh sách block chưa được commit.

Chú ý : ID Block có thể được thấy qua một phần của metadata mà bạn có thể lưu vết cho mỗi block.

Một ví dụ tận dụng ID block là có được ID block ID qua giá trị băm của nội dung dữ liệu qua mỗi block. Trong phương pháp này, đây là một trong những cách có thể kiểm tra tính toàn vẹn dữ liệu của mỗi block dữ liệu nhận được từ hệ thống.

## Blobs – Azure Storage Client v1.0

Mặc dù REST API và các thao tác trong REST API được đọc một cách dễ dàng. Nhưng để thuận tiện hơn, nhóm Windows Azure SDK đã tạo ra thư viện hỗ trợ: Microsoft.WindowsAzure.StorageClient. Thư viện này được sử dụng để gọi REST API của dịch vụ lưu trữ Windows Azure.

Azure Storage Client v1.0 cung cấp các lớp hỗ trợ chức năng Azure Blob với những lớp chính sau:

* CloudStorageAccount
* CloudBlob
* CloudBlobClient
* CloudBlobContainer
* CloudBlobDirectory
* CloudBlockBlob
* CloudPageBlob

*CloudStorageAccount* Là một lớp hỗ trợ lấy thông tin tài khoản từ tập tin cấu hình hoặc tạo ra một thực thể của đối tượng tài khoản lưu trữ từ các thông số.

*CloudBlob* là lớp cơ sở cho *CloudBlobDirectory, CloudBlockBlob* và *CloudPageBlob.*Nó cung cấp hầu hết các chức năng để xử lí blob bao gồm : tạo và xóa blob; và duy trì các đặc tính blob bao gồm thuộc tính và metadata.

*CloudBlobDirectory* thêm các chức năng truy xuất blob với các tên được cấu trúc như hệ thống cây thư mục.

*CloudBlockBlob* thêm các chức năng đặc trưng cho Block Blob trong khi *CloudPageBlob* thêm các chức năng đặc trưng cho Page Blob.

*CloudBlobClient* cung cấp chức năng lấy danh sách các blob và tăng thêm truy vấn trực tiếp đến các kiểu blob khác nhau. Nó cũng cung cấp cách thực thuận tiện để kết nối với đối tượng CloudStorageAccount cung cấp quyền truy xuất đến các thông tin xác thực được lưu trữ trong cấu hình Azure. *CloudBlobContainer* cung cấp các chức năng để quản lí container bao gồm tạo, xóa và lấy danh sách.

Ngoài lớp chính trên, còn có những lớp khác như BlobProperty và BlobContainerProperties, CloudPageBlob và CloudBlockBlob…

Các bước để lập trình một ứng dụng Blob đơn giản với các lớp trên như sau:

1. Thêm lệnh sau vào lớp C# của bạn:

*using Microsoft.WindowsAzure.StorageClient;*

2. Tạo một đối tượng lớp CloudStorageAccount từ tập tin cấu hình:

*CloudStorageAccount storageAccountInfo =*

*CloudStorageAccount.FromConfigurationSetting(configurationSettingName);*

3. Hoặc, tạo một đối tượng lớp CloudStorageAccount sử dụng thông tin tài khoản:

*CloudStorageAccount storageAccountInfo = new CloudStorageAccount(new*

*StorageCredentialsAccountAndKey(accountName, accountKey), new Uri(blobEndpointURI), new*

*Uri(queueEndpointURI), new Uri(tableEndpointURI));*

4. Tạo một đối tượng CloudBlobClient:

*CloudBlobClient blobStorageType = storageAccountInfo.CreateCloudBlobClient();*

5. Khi bạn có một đối tượng lớp CloudBlobClient, bạn có thể xử lí các thao tác trên dịch vụ Blob Storage:

Lấy container :

*IList<CloudBlobContainer> containers = new*

*List<CloudBlobContainer>(this.blobStorageType.ListContainers(prefix,*

*ContainerListingDetails.All));*

Tạo một container:

*blobStorageType.GetContainerReference(containerName).CreateIfNotExist();*

Tạo một container với các quyền:

*CloudBlobContainer container = blobStorageType.GetContainerReference(containerName);*

*BlobContainerPermissions perm = new BlobContainerPermissions();*

*perm.PublicAccess = accessType;*

*container.SetPermissions(perm);*

*container.Metadata.Add(new NameValueCollection());*

*container.CreateIfNotExist();*

Tạo một blob bằng cách tải lên các mảng byte:

*blobStorageType.GetContainerReference(containerName).GetBlobReference(blobName).*

*UploadByteArray(blobContents);*

Tạo một blob bằng cách tải lên text:

*blobStorageType.GetContainerReference(containerName).GetBlobReference(blobName)*

*.UploadText(blobContents);*

Tạo một blob bằng cách tải lên một stream:

*blobStorageType.GetContainerReference(containerName).GetBlobReference(blobName)*

*.UploadFromStream(blobContents);*

Tạo một blob bằng cách tải lên một tập tin:

*blobStorageType.GetContainerReference(containerName).GetBlobReference(blobName).*

*UploadFile(fileName);*

Lấy một blob bằng cách tải xuống một mảng byte:

*blobStorageType.GetContainerReference(containerName).GetBlobReference(blobName).*

*DownloadByteArray();*

Lấy một blob bằng cách tải xuống text:

*blobStorageType.GetContainerReference(containerName).GetBlobReference(blobName).*

*DownloadText();*

Lấy một blob bằng cách tải xuống một stream:

*blobStorageType.GetContainerReference(containerName).GetBlobReference(blobName).*

*DownloadToStream(outputStream);*

Lấy một blob bằng cách tải xuống một file:

*blobStorageType.GetContainerReference(containerName).GetBlobReference(blobName).*

*DownloadToFile(outputFileName);*

### 

### CloudBlobClient

Đối tượng CloudBlobClient có thể được tạo hoặc sử dụng một trong các hàm contructor hoặc gọi phương thức mở rộng :

*public static CloudBlobClient CreateCloudBlobClient(CloudStorageAccount account);*

Ví dụ :

*CloudStorageAccount cloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
CloudBlobClient cloudBlobClient = new CloudBlobClient(cloudStorageAccount.BlobEndpoint.ToString(), cloudStorageAccount.Credentials);*

Hoặc

*CloudStorageAccount cloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

**CloudBlobClient** cung cấp nhiều phương thức đồng bộ để lấy danh sách các container, cũng như các phương thức không đồng bộ mà còn hỗ trợ phân trang theo danh sách. Chú ý hậu tố Segmented được sử dụng suốt Azure Storage Client để hỗ trợ phân trang. Đối tượng ResultSegment<CloudBlobContainer> được trả về từ phương thức EndListContainersSegmented(). Lớp ResultSegment<TElement> cung cấp các phương thức

BeginGetNext() và EndGetNext() có thể được sử dụng đánh số trang qua danh sách các danh sách các container. Phương thức ListBlobWithPrefix() lấy danh sách các container trong container $root. GetContainerReference() cung cấp tham chiếu đến một container.

Ví dụ sau cho thấy cách sử dụng của GetBlockBlob():

*protected void GetBlockBlobStream(String blobAddress)  
{  
    CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlockBlob cloudBlockBlob = cloudBlobClient.GetBlockBlob(blobAddress);*

*using( MemoryStream memoryStream = new MemoryStream())  
    {  
        cloudBlockBlob.DownloadToStream(memoryStream);  
        Int64 streamSize = memoryStream.Length;  
        memoryStream.Seek(0, SeekOrigin.Begin);  
        using (StreamReader streamReader = new StreamReader(memoryStream))  
        {  
            String blobText = streamReader.ReadToEnd();  
        }  
    }  
}*

Ví dụ tạo ra một tham chiếu đến đối tượng CloudBlockBlob và sau đó tải vào [MemoryStream](http://msdn.microsoft.com/en-us/library/system.io.memorystream.aspx). Chú ý rằng trước đó memoryStream đã được sử dụng phương thức Seek() để thiết lập vị trí hiện tại ở đầu stream. Cho một container có tên music chứa một blob có tên British/StoneRoses ,phương thức có thể được gọi :

*GetBlockBlobStream("music/British/StoneRoses")*

### CloudBlobContainer

Lớp CloudBlobContainer đại diện một container blob. Nó cung cấp các phương thức hỗ trợ tạo và xóa container cũng như cung cấp các chức năng bảo trì như gán và lấy các thuộc tính, metadata và thiết lập bảo mật cho container. Lớp CloudBlobContainer cũng cung cấp các phương thức cho phép truy xuất trực tiếp.

Một CloudBlobContainer được tạo bằng cách sử dụng một trong các constructor và được cung cấp tên container. Ví dụ :

*protected void CreateContainer(String containerName)  
{  
    CloudStorageAccount cloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = new CloudBlobContainer(containerName, cloudBlobClient);  
    cloudBlobContainer.Metadata["MetadataName"] = "MetadataValue";  
    cloudBlobContainer.Create();  
}*

ListBlobs() có thể được sử dụng để liệt kê các blob trong container :

*protected void GetBlobs(String containerName)  
{  
    CloudStorageAccount cloudStorageAccount = CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName);*

*IEnumerable<IListBlobItem> blobList = cloudBlobContainer.ListBlobs();  
    foreach (IListBlobItem item in blobList)  
    {  
        Uri uri = item.Uri;  
    }  
}*

Thuộc tính container có thể được gán bằng cách sử dụng thuộc tính [Attributes](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblobcontainer.attributes.aspx). Metadata có thể được gán bằng cách sử dụng [Metadata](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblobcontainer.metadata.aspx) [NameValueCollection](http://msdn.microsoft.com/en-us/library/system.collections.specialized.namevaluecollection.aspx) như ví dụ CreateContainer() ở trên. [SetMetadata()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblobcontainer.setmetadata.aspx) có thể được sử dụng để cập nhật Metadata trong một container.

Không có phương thức để kiểm tra một container có tồn tại hay không.

### CloudBlob

Lớp [CloudBlob](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblob.aspx) đại diện một blob và thực thi các chức năng không định rõ cho các lớp đại diện cho block blob và page blob.

##### Blob Addressing

Một blob tồn tại trong một thùng chứ và một phương pháp đơn giản nhất để đánh địa chỉ một blob là sử dụng mẫu containername/blobname, với một URL đầy đủ:

https://myaccount.blob.core.windows.net/containername/blobname

Tuy nhiên tên blob có thể bao gồm dấu phân cách, tiêu biểu như ‘/’, có thể được sử dụng để mô phỏng sự phân cấp cây thư mục trong namespace của các blob. Ví dụ, dưới đây là tất cả các địa chỉ hợp lệ cho blob trong container được đặt tên *music*:

* *music/American/BobDylan/PositivelyFourthStreet*
* *music/American/StevieWonder/InnerVisions*
* *music/British/Portishead/SourTimes*
* *music/British/Portishead/WanderingStar*

Giản đồ tên thư mục mô phỏng American và British như thư mục cấp cao nhất với BobDylan, StevieWonder và Portishead là thư mục cấp thấp. Thư mục *Portishead* chứa hai blob được xác định bởi SourTimes và WanderingStar mặc dù tên đầy đủ của chúng là British/Portishead/SourTimes và British/Portishead/WanderingStar. Khi gọi

*GetDirectoryList(“music/British”, “Portishead”);*

Mã bên dưới mô tả cách liệt kê hai blob này:

*protected void GetDirectoryList(String topLevelDirectoryName, String subDirectoryName)  
{  
    CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");*

*CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobDirectory topLevelDirectory = cloudBlobClient.GetBlobDirectoryReference(topLevelDirectoryName);*

*CloudBlobDirectory subDirectory = topLevelDirectory.GetSubdirectory(subDirectoryName);*

*IEnumerable<IListBlobItem> blobItems = subDirectory.ListBlobs();  
    foreach (IListBlobItem blobItem in blobItems)  
    {  
        Uri uri = blobItem.Uri;  
    }  
}*

Đối tượng CloudBlobClient có thể được tạo ra sử dụng một trong các hàm constructor hoặc gọi phương thức CloudBlobClient:

*public CloudBlob GetBlobReference(String blobAddress);*

Các hàm này không tạo ra một blob trong dịch vụ lưu trữ Azure mà chỉ là một đại diện trong bộ nhớ. Việc tạo ra thực tế đòi hỏi tải lên nội dung blob thông qua một trong những phương pháp tải lên. Ví dụ:

*protected void UploadText(String containerName, String blobName)  
{  
    CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName);*

*CloudBlob cloudBlob = cloudBlobContainer.GetBlobReference(blobName);  
    cloudBlob.Metadata[“MetadataName”] = “MetadataValue”;  
    cloudBlob.UploadText("Stately, plump Buck Mulligan came from the stairhead");  
}*

Một blob có thể được tạo ra trong dịch vụ lưu trữ bằng cách sử dụng [OpenWrite()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblob.openwrite.aspx) để tạo một đối tượng [BlobStream](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.blobstream.aspx) và sau đó viết vào stream:

*protected void WriteToBlobStream(String containerName, String blobName)  
{  
    CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName);*

*CloudBlob cloudBlob = cloudBlobContainer.GetBlobReference(blobName);*

*String blobText = "bearing a bowl of lather on which a mirror and a razor lay crossed";*

*UTF8Encoding utf8Encoding = new UTF8Encoding();  
    Byte[] bytes = utf8Encoding.GetBytes(blobText);*

*using (BlobStream blobStream = cloudBlob.OpenWrite())  
    {  
        blobStream.Write(bytes, 0, bytes.Count<Byte>());  
    }*}

Khi BlobStream được đóng, nó tải các block và sau đó commit các block.

Nội dung của blob có thể được lấy sử dụng một trong những phương pháp tải về hoặc sử dụng [OpenRead()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblob.openread.aspx) để tạo một đối tượng BlobStream và sau đó đọc từ stream. Ví dụ:

*protected void DownloadText(String containerName, String blobName)  
{  
    CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName);  
    CloudBlob cloudBlob = cloudBlobContainer.GetBlobReference(blobName);  
    String blobText = cloudBlob.DownloadText();  
}*

Và :

*protected void ReadBlobStream(String containerName, String blobName)  
{  
    CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName);  
    CloudBlob cloudBlob = cloudBlobContainer.GetBlobReference(blobName);*

*Byte[] bytes = new Byte[1000];  
    using ( BlobStream blobStream = cloudBlob.OpenRead())  
    {  
        blobStream.Read(bytes, 0, bytes.Count<Byte>());  
    }  
}*

Blob có thể được xóa khỏi Dịch vụ Lưu trữ Azure với phương thức Delete() và [DeleteIfExists()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblob.deleteifexists.aspx). Sau đó là một số phương thức thuận tiện xác nhật sự tồn tại của blob trước khi blob được xóa.

Blob có thể được sao chép trực tiếp trên Dịch vụ Lưu trữ Azure sử dụng phương thức [CopyFromBlob()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblob.copyfromblob.aspx)

*protected void CopyFromBlob(String containerName, String blobName)  
{  
    CloudStorageAccount cloudStorageAccount =  
      CloudStorageAccount.FromConfigurationSetting("DataConnectionString");*

*CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = CloudBlobClient.GetContainerReference(containerName);*

*CloudBlockBlob cloudBlockBlob = cloudBlobContainer.GetBlockBlobReference(blobName);*

*CloudBlockBlob copyCloudBlockBlob = cloudBlobContainer.GetBlockBlobReference("Copy" + blobName);*

*copyCloudBlockBlob.CopyFromBlob(cloudBlockBlob);  
}*

### CloudBlockBlock

Lớp [CloudBlockBlob](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudblockblob.aspx) có nguồn gốc từ lớp CloudBlob và thêm các chức năng riêng biệt cho block blob.

Chức năng CloudBlockBlob được cung cấp cả dạng đồng bộ và không đồng bộ.

Block blob bao gồm một chuỗi liên tục các block được tải lên một cách động lập nhờ phương thức PutBlock() và sau đó commit vào blob sử dụng PutBlockList(). Các Block được xác định bằng block ID. Danh sách block có thể được hiển thị sử dụng DownloadBlockList(). Phần code dưới đây cho thấy một block đơn được tải lên blob qua [MemoryStream](http://msdn.microsoft.com/en-us/library/system.io.memorystream.aspx) :

*protected void PutBlockFromStream(String containerName, String blobName, Int32 blockId)  
{*

*CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");*

*CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName)*

*CloudBlockBlob cloudBlockBlob = cloudBlobContainer.GetBlockBlobReference(blobName);*

*String blobText = new String('b', 1000);  
    UTF8Encoding utf8Encoding = new UTF8Encoding();  
    using (MemoryStream memoryStream = new MemoryStream(utf8Encoding.GetBytes(blobText)))  
    {  
        cloudBlockBlob.PutBlock(Convert.ToBase64String(System.BitConverter.GetBytes(blockId)), memoryStream, null);  
    }  
}*

Phần dưới hiển thị danh sách block được commit để tạo một blob block trong Dịch vụ Lưu trữ Azure:

*protected void PutBlockList(String containerName, String blobName, Int32 numBlocks)  
{  
    CloudStorageAccount cloudStorageAccount   
      CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName);  
    CloudBlockBlob cloudBlockBlob = cloudBlobContainer.GetBlockBlobReference(blobName);*

*String[] blockIds = new String[numBlocks];  
    for (Int32 i = 0; i < numBlocks; i++)  
    {  
        blockIds[i] = Convert.ToBase64String(System.BitConverter.GetBytes(i)); ;  
    }*

*cloudBlockBlob.PutBlockList(blockIds);  
}*

Ví dụ này giả định rằng blockID là các con số tuần tự từ 0 đến nhỏ hơn số lượng các block.

### CloudPageBlob

Một page blob bao gồm một dãy tuần tự các page và định địa chỉ cho tối ưu hóa việc đọc và ghi ngẫu nhiên. Cả page và page blob đều có kích thước tối đa là 1 TB. Còn Block blobs được tối ưu hóa cho stream.

Chỉ có một cách viết vào page blob là qua phương thức [WritePages()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudpageblob.writepages.aspx) và phiên bản không đồng bộ của nó [BeginWritePages()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudpageblob.beginwritepages.aspx) và [EndWritePages()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudpageblob.endwritepages.aspx).

Một page blob được tạo trong Dịch vụ Lưu trữ Azure sử dụng phương thức [Create()](http://msdn.microsoft.com/en-us/library/microsoft.windowsazure.storageclient.cloudpageblob.create.aspx). Dữ liệu được viết vào các trang riêng lẻ trong một page blob sử dụng WritePages(). Bất kì phương thức tải dữ liệu của CloudBlob đều có thể sử dụng để lấy dữ liệu từ page blob. Khi tải về, bất kì trang nào không được viết sẽ được thiết lập là 0.

Sau đây cho thấy việc tạo một page blob và tải một vài dữ liệu lên page thứ nhất và thứ ba trong bốn page :

*protected void CreatePageBlob(String containerName, String blobName)  
{  
    CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName);*

*CloudPageBlob cloudPageBlob = cloudBlobContainer.GetPageBlobReference(blobName);*

*const Int32 pageSize = 512;  
    const Int32 numPages = 4;  
    cloudPageBlob.Create(numPages \* pageSize);*

*String blobText = new String('z', pageSize);  
    UTF8Encoding utf8Encoding = new UTF8Encoding();  
    using (MemoryStream memoryStream = new MemoryStream(utf8Encoding.GetBytes(blobText)))  
    {  
        cloudPageBlob.WritePages(memoryStream, 0);  
        memoryStream.Seek(0, SeekOrigin.Begin);  
        cloudPageBlob.WritePages(memoryStream, 2 \* pageSize);  
    }  
}*

Sau đây là lấy danh sách của các dãy page trong page blob cũng như tải dữ liệu page blob vào MemoryStream:

*protected void GetPageBlob(String containerName, String blobName)  
{  
    CloudStorageAccount cloudStorageAccount =  
       CloudStorageAccount.FromConfigurationSetting("DataConnectionString");  
    CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();*

*CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(containerName);*

*CloudPageBlob cloudPageBlob = cloudBlobContainer.GetPageBlobReference(blobName);*

*IEnumerable<PageRange> pageRanges = cloudPageBlob.GetPageRanges();  
    foreach (PageRange pageRange in pageRanges)  
    {  
        Int64 startOffset = pageRange.StartOffset;  
        Int64 endOffset = pageRange.EndOffset;  
    }*

*using (MemoryStream memoryStream = new MemoryStream())  
    {  
        cloudPageBlob.DownloadToStream(memoryStream);  
        Int64 streamSize = memoryStream.Length;  
    }  
}*

## Các chiến lược tải block lên

Tải lên một blob theo danh sách các block chúng ta có được các lợi ích sau:

* Sự nối tiếp – khi mỗi block được tải lên thành công điều này có thể chắc chắn thành công của khối này và thử lại khối nếu có thất bại và tiếp tục từ điểm đó.
* Tải lên song song – điều này có thể tải các block lên song song làm giảm thời gian tải của các blob rất lớn.
* Tải lên không theo thứ tự - Bạn có thể tải các block lên mà không theo thứ tự.
* Kết hợp BlockID như Metadata cho mỗi block.

**PutBlock**(BlockId1);

**PutBlock**(BlockId3);

**PutBlock**(BlockId4);

**PutBlock**(BlockId2);

**PutBlock**(BlockId4);

**PutBlockList**(BlockId2, BlockId3, BlockId4);

Figure 4 Chiến lược tải block lên

Ví dụ hình 4 sẽ được sử dụng để giải thích các chiến lược khác nhau có thể gặp khi sử dụng giao tiếp block để tải lên các blob. Ở đó là:

* Tải lên các block với cùng Block ID – Khi các block với cùng ID được tải lên cùng blob, block được úp lên gần nhất cùng với ID sẽ được sử dụng khi commit blob với PUT blocklist. Trong ví dụ trên, hai block với ID = BlockId4 được tải lên và block sau cùng sẽ được sử dụng trong block list commit cho blob.
* Tải các block lên không theo thứ tự - Block có thể được tải lên theo thứ tự khác nhau như trong danh sách block tải lên blob. Trong ví dụ ở trên, danh sách block được commit cuối cùng là những block theo thứ tự BlockId2, BlockId3, BlockId4, nhưng những block này được upload theo thứ tự khác nhau. Dữ liệu blob có thể đọc được (qua GET) được sắp xếp liên quan đến danh sách được đặt tả trong PUT blocklist. Do đó, nếu bạn lấy một blob và đọc blob từ lúc bắt đầu đến khi kết thúc bạn sẽ thấy nội dung của BlockId2, theo sau la BlockId3, sau nữa là Block ID 4.

Các Block không sử dụng – Hơn nữa, một vài block có thể không bao giờ lấy được commit vào danh sách block cuối cùng trong blob. Các block này sẽ được dọn dẹp bởi hệ thống. BlockId1 và Block với ID BlockId4 đầu tiên trong ví dụ sẽ được thu dọn.

#### 3.1.4.3. Table

Windows Azure Table là bộ lưu trữ có cấu trúc được cung cấp bởi nền tảng Windows Azure. Nó hỗ trợ các bảng có khả năng mở rộng cực lớn trên đám mây, nó có thể chứa hàng tỉ thực thể và terabyte dữ liệu. Hệ thống sẽ tự động mở rộng một cách hiệu quả đến hàng ngàn máy chủ.

Bộ lưu trữ có cấu trúc được cung cấp dưới dạng bảng, mỗi bảng chứa một tập các thực thể, mỗi thực thể chứ một tập các thuộc tính được đặt tên. Một số điểm nổi bật của Windows Azure Table :

* Hỗ trợ LINQ, dịch vụ dữ liệu ADO .NET và REST.
* Tập hợp các kiểu dữ liệu phong phú cho các giá trị thuộc tính.
* Hỗ trợ số lượng bảng và thực thể không giới hạn và không bị giới hạn về kích thước bảng.
* Dùng mô hình optimistic concurrency cho những thao tác cập nhật và xóa.
* Đối với các truy vấn lấy ra số lượng lớn các kết quả hoặc truy vấn bị ngắt quãng. Từng phần kết quả được trả về với một cái thẻ nối tiếp cho phép truy vấn lại tiếp tục tại nơi mà nó đã dừng lại.

# Mô hình dữ liệu bảng

Sau đây tóm tắt các mô hình dữ liệu cho Windows Azure Table:

* **Storage Account (Tài khoản lưu trữ) -** Ứng dụng phải sử dụng một tài khoản hợp lệ để truy xuất bộ lưu trữ Windows Azure. Bạn có thể tạo một tài khoản mới qua trang web của Windows Azure. Người dùng sẽ nhận được một khóa bí mật 256-bit cho mỗi tài khoản được tạo. Khóa bí mật này dùng để chứng thực những truy vấn từ người dùng đến hệ thống lưu trữ. Đặc biệt, một chữ kí HMAC SHA256 cho mỗi request sử dụng khóa bí mật này. Chữ ký được chuyển qua với mỗi request để xác thực các request của người dùng.

Tên tài khoản là một phần của hostname trên đường dẫn URL. Hostname để truy xuất các bảng là <accountName>.table.core.windows.net.

* **Table (Bảng)** – chứa một bộ các thực thể. Một ứng dụng có thể tạo ra nhiều bảng chỉ với một tài khoản lưu trữ.
* **Entity (Thực thể)** – Thực thể (một thực thể tương tự như một dòng) là mẫu dữ liệu cơ bản được lưu trữ trong một bảng. Một thực thể chứa một tập các thuộc tính. Mỗi bảng có 2 thuộc tính tên là “PartitionKey” và “RowKey” tạo ra khóa duy nhất cho thực thể.
* **Property (Thuộc tính)** – Nó đại diện cho một giá trị duy nhất trong thực thể. Tên thuộc tính phân biệt hoa, thường. Một tập hợp phong phú các kiểu dữ liệu được hỗ trợ cho giá trị thuộc tính.
* **PartitionKey** – Đây là thuộc tính khóa đầu tiên của mỗi bảng. Hệ thống sử dụng khóa này để tự động cấp phát các thực thể của bảng qua nhiều nút lưu trữ.
* **RowKey** – là thuộc tính khóa thứ hai của bảng. Đây là ID duy nhất của thực thể trong phân vùng mà nó thuộc về. PartitionKey kết hợp với RowKey xác định duy nhất một thực thể trong một bảng.
* **Timestamp** – là thuộc tính có giá trị **Datetime**, được lưu giữ bởi hệ thống để ghi lại thời điểm thực thể thay đổi cuối cùng.
* **Partition** – Một tập hợp các thực thể trong một bảng với cùng giá trị khóa PartitionKey.
* **Sort Order (Thứ tự sắp xếp)** – Các thực thể trong bảng được sắp xếp theo PartitionKey vào sau đó là theo RowKey. Điều này có nghĩa là các truy vấn xác định theo các khóa này sẽ hiệu quả hơn, và tất cả các kết quả được trả về được sắp xếp theo PartitionKey và sau đó theo RowKey.

Một bảng có một giản đồ linh hoạt. Windows Azure Table lưu giữ tên và kiểu dữ liệu cho mỗi thuộc tính trong một thực thể.

Sau đâu là một số thông tin chi tiết bổ sung cho Bảng, Thực thể và Thuộc tính :

* Bảng
  + Tên bảng chỉ chứa các kí tự và số.
  + Tên bảng không bắt đầu với một kí tự số.
  + Tên bảng phân biệt hoa – thường.
  + Tên bảng phải có độ dài từ 3 đến 63 kí tự.
* Tên thuộc tính

Chỉ được chứa các kí tự chữ cái và dấu ‘\_’.

* Một thực thể có thể có tối đa 255 thuộc tính bao gồm các thuộc tính bắt buộc - PartitionKey, RowKey và Timestamp. Tất cả các thuộc tính khác trong thực thể có tên được định nghĩa theo nhu cầu ứng dụng.
* PartitionKey và RowKey có kiểu chuỗi, và mỗi khóa bị giới hạn trong kích thước 1 KB.
* Timestamp là thuộc tính chỉ đọc.
* Không có giản đồ cố định – Không có giản đồ được lưu trữ bởi Windows Azure Table, vì thế tất cả các thuộc tính được lưu trữ theo cặp <tên, kiểu dữ liệu>. Điều này có nghĩa rằng 2 thực thể trong cùng một bảng có thể có thuộc tính khác nhau. Một bảng có thể có hai thực thể với tên thuộc tính giống nhau, nhưng có kiểu dữ liệu khác nhau cho giá trị thuộc tính. Tuy nhiên, tên thuộc tính phải là duy nhất với mỗi thực thể.

• Kết hợp kích thước của tất cả các dữ liệu trong một thực thể không thể vượt quá 1MB. Kích thước này bao gồm kích thước của tên thuộc tính cũng như kích thước của giá trị thuộc tính hoặc kiểu của chúng, bao gồm cả hai thuộc tính khoá bắt buộc (PartitionKey và RowKey).

* Hỗ trợ các kiểu thuộc tính: Binary, Bool, DateTime, Double, GUID, Int, Int64, String.

Xem bảng bên dưới để biết giới hạn.

| **Kiểu thuộc tính** |  |
| --- | --- |
| Binary | Kiểu mảng các byte 64 KB. |
| Bool | Kiểu Boolean. |
| DateTime | Kiểu thời gian 64-bit. Miền giá trị được hỗ trợ từ ngày 1/1/1601 đến 12/31/9999. |
| Double | Kiểu số thực 64-bit. |
| GUID | Kiểu dữ liệu để chỉ định ID 128-bit. |
| Int | Kiểu số nguyên 32-bit. |
| Int64 | Kiểu số nguyên 64-bit. |
| String | Kiểu chuỗi. Giá trị chuỗi có thể lên đến 64 KB. |

# Phân vùng bảng

Windows Azure Table cho phép các bảng mở rộng đến hàng ngàn nút lưu trữ bằng cách phân phối các thực thể trong bảng. Khi phân phối các thực thể, nó mong muốn đảm bảo một tập hợp các thực thể luôn nằm cùng trên một nút lưu trữ. Một ứng dụng điều chỉnh thiết lập này bằng cách chọn giá trị thích hợp cho thuộc tính PartitionKey trong mỗi thực thể.

Hình minh họa 1 - Ví dụ phân vùng

Hình mình họa mô tả một bảng chứa nhiều phiên bản của nhiều tài liệu khác nhau. Mỗi thực thể trong bảng tương ứng với một phiên bản của một tài liệu cụ thể. Trong ví dụ này, PartitionKey của bảng là tên tài liệu, và RowKey là tên phiên bản. Tên tài liệu cùng với tên phiên bản xác định duy nhất một thực thể trong bảng. Trong ví dụ này, tất cả các phiên bản của cùng một tài liệu tổ chức vào một phân vùng riêng lẻ.

## Tác động của việc phân vùng

Bây giờ chúng ta mô tả mục đích của các phân vùng và làm thế nào để chọn một PartitionKey.

### Khả năng mở rộng của bảng

Hệ thống lưu trữ đạt được khả năng mở rộng khá lớn bằng cách phân bố các phân vùng qua nhiều nút lưu trữ. Hệ thống quan sát mô thức sử dụng các phân vùng, và tự động cân bằng các phân vùng này qua tất cả các nút lưu trữ. Điều này cho phép hệ thống và ứng dụng của bạn mở rộng để đáp ứng nhu cầu lưu thông của các bảng. Nghĩa là, nếu có nhiều lưu thông đến một vài phân vùng, hệ thống sẽ tự động lan chúng ra đến nhiều nút lưu trữ, vì thế tải lưu thông sẽ được lan ra qua nhiều máy chủ. Tuy nhiên, một phân vùng ,có nghĩa là với tất cả các thực thể có cùng PartitionKey, sẽ được đáp ứng bởi một nút lưu trữ. Tuy nhiên, lượng dữ liệu được lưu trữ trong một phân vùng không bị giới hạn bởi dung lượng lưu trữ của một nút.

### Giao dịch nhóm thực thể

Đối với các thực thể được lưu trữ trong cùng một bảng và cùng một phân vùng, ứng dụng có thể thực hiện một giao dịch bao hàm nhiều thực thể. Điều này cho phép giao dịch thực hiện nhiều thao tác Thêm/ Xóa/ Sửa qua nhiều thực thể với chỉ một request batch đến hệ thống lưu trữ, với điều kiện tất cả các thực thể ở cùng phân vùng trong một bảng. Hoặc tất cả các thao tác thực hiện thành công trong một giao dịch hoặc chúng bị thất bại hoàn toàn.

### Miền thực thể

Các thực thể trong một phân vùng được lưu trữ cùng nhau. Điều này cho phép truy vấn hiệu quả trong một phân vùng.

Trong ví dụ trên, tất cả các phiên bản của cùng một tài liệu tổ chức trong một phân vùng riêng. Do đó, việc truy xuất “tất cả các phiên bản của một tài liệu xác định” sẽ đạt hiệu quả cao, bởi vì chúng ta truy xuất đến một phân vùng duy nhất. Mặt khác, một truy vấn “tất cả các phiên bản của các tài liệu được chỉnh sửa trước 5/30/2007” không bị giới hạn ở một phân vùng duy nhất. Bởi vì truy kiểm tra tất cả các phân vùng trên nhiều nút lưu trữ, một truy vấn như vậy sẽ tốn nhiều chi phí hơn.

## Chọn một PartitionKey

Việc lựa chọn một PartitionKey đóng vai trò quan trọng để một ứng dụng có khả năng mở rộng tốt. Có một sự đánh đổi ở đây giữa việc cố gắng để được lợi từ miền thực thể, ở đó bạn đạt được những truy vấn hiệu quả qua các thực thể trong cùng một phân vùng, và khả năng mở rộng của các bảng, ở đây càng nhiều phân vùng trong bảng thì càng dễ dàng ở cho Windows Azure Table lan truyền tải qua nhiều máy chủ.

### Giao dịch nhóm thực thể

Nếu ứng dụng của bạn cần sử dụng giao dịch nhóm thực thể, một PartitionKey cần phải được lựa chọn để qui mô của nó có thể bao trùm hết tất cả các thực thể bạn cần thực hiện giao dịch trên đó. Tùy theo nhu cầu truy vấn của bạn, cần chọn PartitionKey mà nó chỉ nhóm cùng các thực thể cần được nhóm lại với nhau, để thực hiện giao dịch nhóm thực thể trên chúng.

### Truy vấn hiệu quả

Chúng ta nhận thấy rằng truy vấn có tần suất cao sử dụng PartitionKey như một điều kiện lọc truy vấn. Sử dụng ParititionKey trong bộ lọc truy vấn hạn chế thực thi truy vấn đến một phân vùng đơn hoặc thành phần con của phân cùng (tùy thuộc vào điều kiện sử dụng), do đó cải thiện được hiệu suất truy vấn.

Nếu PartitionKey không phải là một phần của truy vấn, thì truy vấn phải đi qua tất cả các phân vùng trong bảng để tìm ra thực thể cần tìm, điều này thực sự không mang lại hiệu quả.

Sau đây là một số hướng dẫn cơ bản và các đề xuất cho cách chọn một PartitionKey để bảng của bạn cho hiệu quả truy vấn cao:

1. Đầu tiên xác định các thuộc tính quan trọng trong bảng của bạn. Đây là những thuộc tính thường xuyên sử dụng như bộ lọc truy vấn.
2. Lựa chọn những khóa có khả năng từ những thuộc tính quan trọng này.
   1. Điều quan trọng là xác định các truy vấn chi phối đến khối lượng công việc của ứng dụng. Từ những truy vấn chi phối này, lựa chọn các thuộc tính được sử dụng trong bộ lọc truy vấn.
   2. Đây là tập hợp các thuộc tính khóa ban đầu của bạn.
   3. Sắp xếp các thuộc tính khóa theo tầm quan trọng trong truy vấn của bạn.
3. Các thuộc tính khóa này xác định độc nhất các thực thể chưa ? Nếu chưa, thêm định danh duy nhất (ID) vào tập hợp các khóa.
4. Nếu bạn chỉ có một thuộc tính khóa, sử dụng nó làm PartitionKey.
5. Nếu bạn chỉ có 2 thuộc tính khóa, sử dụng thuộc tính đầu tiên làm ParitionKey, và thuộc tính thứ 2 làm RowKey.
6. Nếu bạn có hơn 2 thuộc tính khóa, bạn có thể ghép chúng thành hai nhóm – nhóm ghép đầu tiên là PartitionKey, và nhóm thứ hai là RowKey. Với PartitionKey có thể chứa hai khóa cách nhau bởi dấu “-“.

### Khả năng mở rộng

Bây giờ ứng dụng có tập hợp các khóa tiềm năng của nó, bạn cần chắc chắn rằng phân vùng đã chọn có khả năng mở rộng :

1. Dựa vào PartitionKey trên, nó có dẫn đến những phân vùng có thể trở nên quá nóng khi đáp ứng từ một máy chủ đơn lẻ không? Một cách để xác định điều này bằng cách thực hiện kiểm thử “stress test Table partition”. Để thực hiện kiểm thử này, cần tạo ra một bảng tương tự sử dụng những khóa của bạn và sau đó gây ứng suất lớn đến khối lượng công việc của bạn lên một phân vùng đơn lẻ để đảm bảo rằng phân vùng có thể đáp ứng thông lượng mong muốn cho ứng dụng của bạn.
2. Nếu kiểm thử “stress test Table partition” thành công, bạn đã hoàn thành.
3. Nếu kiểm thử “stress test Table partition” không thành công, lựa chọn một PartitionKey có tính chất tốt hơn. Điều này có thể được thực hiện bằng cách chọn một PartitionKey khác hoặc chỉnh sửa PartitionKey hiện tại (ví dụ, bằng cách liên kết nó với thuộc tính khóa tiếp theo). Mục đích của điều này là tạo ra nhiều phân vùng để một phân vùng riêng lẻ không trở nên quá lớn hoặc quá nóng.
4. Chúng ta đã thiết kế hệ thống có thể mở rộng và có khả năng xử lí một khối lượng lưu thông lớn. Tuy nhiên với tần suất truy vấn cực lớn có thể dẫn đến lỗi truy vấn. Trong trường hợp này, cần giảm tần suất truy vấn của bạn hoặc khử lỗi.

# Lập trình Bảng

Các hoạt động cơ bản sau đây được hỗ trợ trên các bảng và các thực thể

* Tạo bảng hoặc thực thể.
* Lấy một bảng hoặc thực thể, với bộ lọc.
* Cập nhật một thực thể (không có cập nhật bảng).
* Xóa một bảng hoặc thực thể.
* Giao dịch nhóm thực thể - hỗ trợ các giao dịch qua các thực thể trong cùng một bảng và cùng một phân vùng.

Để sử dụng bảng trong ứng dụng .Net, bạn có thể sử dụng dịch vụ dữ liệu ADO.NET.

Bảng phía dưới tóm tắt các API. Bởi vì API dịch vụ dữ liệu.NET ADO.NET thực hiện truyền các gói tin REST, nên ứng dụng có thể chọn sử dụng trực tiếp REST.

| **Nghiệp vụ** | **ADO.NET Data Services** | **HTTP Verb** | **Tài nguyên** | **Mô tả** |
| --- | --- | --- | --- | --- |
| Truy vấn | LINQ Query | GET | Bảng | Trả về danh sách các bảng trong tài khoản lưu trữ. Nếu một bộ lọc hiện có, nó sẽ trả về các bảng tương ứng với bộ lọc này. |
| Thực thể | Trả về tất cả các thực thể trong một bảng xác định hoặc tập con các thực thể nếu các tiêu chuẩn lọc được xác định. |
| Cập nhật toàn bộ thực thể | UpdateObject & SaveChanges(SaveChangesOptions.ReplaceOnUpdate) | PUT | Thực thể | Cập nhật giá trị thuộc tính trong một thực thể. Thao tác PUT thay thế toàn bộ thực thể và có thể được sử dụng để thay thế các thuộc. |
| Cập nhật một phần thực thể | UpdateObject & SaveChanges() | MERGE | Thực thể | Cập nhật giá trị thuộc tính với một thực thể. |
| Tạo một thực thể mới | AddObject & SaveChanges() | POST | Bảng | Tạo một bảng mới trong tài khoản lưu trữ. |
| Thực thể | Thêm một thực thể mới vào bảng. |
| Xóa thực thể | DeleteObject & SaveChanges() | DELETE | Bảng | Xóa một bảng trong tài khoản lưu trữ. |
| Thực thể | Xóa một thực thể từ bảng. |
| Giao dịch nhóm thực thể. | SaveChanges(SaveChangesOptions.Batch) | POST | $batch | Hỗ trợ giao dịch nhóm thực thể được cung cấp thông qua một loạt hoạt động trên các thực thể có cùng partition key trong một bảng duy nhất. Trong ADO.NET Data Services, tùy chọn các lệnh SavaChanges. In ADO.NET Data Services, the option to SaveChanges dictates that the request needs to be sent as a single transaction. |

Các hoạt động cấp cao sẽ được thảo luận chi tiết hơn ở phần sau như :

* Phân trang.
* Xử lí xung đột do truy xuất đồng thời.

## Đặt tên phiên bản

Với tất cả các giải pháp của bộ lưu trữ Windows Azure, chúng tôi đã giới thiệu header HTTP mới gọi được gọi là “x-ms-version”. Tất cả những thay đổi đến API sẽ được đặt tên phiên bản trên header này. Điều này cho phép các phiên bản trước của các lệnh xử lí trên hệ thống lưu trữ tiếp tục làm việc, và chúng ta mở rộng khả năng của các lệnh hiện tại và giới thiệu những lệnh mới.

x-ms-version nên được chỉ rõ cho tất cả các request đến bộ lưu trữ Windows Azure. Nếu có một request anonymous mà không có phiên bản, thì phiên bản cũ nhất của lệnh sẽ được xử lí trên hệ thống lưu trữ.

Phiên bản được hỗ trợ hiện tại là “x-ms-version: 2009-04-14”. Phiên bản này có thể được sử dụng cho tất cả các lệnh và request gửi đến bộ lưu trữ Windows Azure.

// thêm header phiên bản sử dụng sự kiện SendingRequest.

context.SendingRequest +=

new EventHandler<SendingRequestEventArgs>(

delegate(object sender, SendingRequestEventArgs requestArgs)

{

HttpWebRequest request = requestArgs.Request as HttpWebRequest;

request.Headers.Add(

"x-ms-date",

DateTime.UtcNow.ToString("R", CultureInfo.InvariantCulture));

request.Headers.Add("x-ms-version", "2009-04-14");

// ... add authorization header using shared key lite

});

## Chạy thử ví dụ

Trong ví dụ dưới đây, chúng tôi mô tả các hoạt động trên bảng “Blogs”. Bảng này được sử dụng để chứa các blog trong ứng dụng MicroBlogging.

Ứng dụng MicroBlogging có hai bảng - Channels và Blogs. Có danh sách các kênh, và các blog được đưa lên một kênh riêng. Cho ứng dụng này, người dùng sẽ đăng ký vào các kênh và họ nhận được các blog mới cho những kênh này mỗi ngày.

Trong ví dụ này, chúng tôi chỉ tập trung vào các bảng blogs, và ví dụ về các bước sau cho bảng Blog:

* + 1. Định nghĩa giản đồ của bảng.
    2. Tạo bảng.
    3. Thêm blog vào bảng.
    4. Lấy danh sách các blog từ bảng.
    5. Cập nhật blog trong bảng.
    6. Xóa blog từ bảng.
    7. Thêm nhiều Blog vào một bảng.

## Định nghĩa lớp thực thể cho bảng

Giản đồ cho một bảng được định nghĩa qua một lớp C#. Đây là mô hình sử dụng dịch vụ dữ liệu ADO.NET.

Định nghĩa thực thể cho các thực thể Blog sẽ được lưu trong bảng Blog. Mỗi thực thể blog có các thông tin sau :

1. ChannelName - Tên kênh (blog được gửi lên kênh này).
2. The posted date – Ngày gửi.
3. Text – Nội dung của blog.
4. Rating – Độ yêu mến của blog này.

Trong bảng “Blogs”, chúng ta chọn tên kênh làm PartitionKey và ngày gửi làm RowKey. PartitionKey và RowKey là những cái khóa của bảng “Blogs” và điều này được chỉ định bằng cách khai báo các khóa trong thuộc tính của lớp “DataServiceKey”. Bảng “Blogs” được phân chia bởi ChannelName. Điều này cho phép các ứng dụng lấy hiệu quả các blog mới nhất cho một kênh mà người dùng đã đăng ký. Thêm vào các khóa, các thuộc tính đặc trưng của người dùng được khai báo như một thuộc tính. Tất cả các thuộc tính với hàm get và set được public được lưu trữ trong bảng Windows Azure. Vì thế, trong ví dụ bên dưới:

* Text và Rating được lưu trữ cho những đối tượng thực thể trong bảng Azure.
* RatingAsString thì không được lưu trữ bởi vì nó không có định nghĩa làm set.
* ID thì không được lưu trữ vì các hàm truy xuất không được public.

[DataServiceKey("PartitionKey", "RowKey")]

public class Blog

{

// ChannelName

public string PartitionKey { get; set; }

// PostedDate

public string RowKey { get; set; }

// User defined properties

public string Text { get; set; }

public int Rating { get; set; }

public string RatingAsString { get; }

protected string Id { get; set; }

}

## Tạo bảng

Tiếp theo, chúng ta thấy làm thế nào để tạo ra những bảng “Blogs” cho tài khoản lưu trữ của bạn. Tạo ra một bảng cũng giống như tạo một thực thể trong một bảng chính gọi là “Tables”. Mọi tài khoản lưu trữ có một bảng chính đã được định nghĩa, và mọi bảng được sử dụng bởi một tài khoản lưu trữ phải đăng kí tên bảng với bảng chính này. Định nghĩa lớp cho bảng chính này được trình bày dưới đây, thuộc tính TableName đại diện cho tên của bảng được tạo ra.

[DataServiceKey("TableName")]

public class TableStorageTable

{

public string TableName { get; set; }

}

Việc tạo ra một bảng được thực hiện như sau.

// Service Uri is “http://<Account>.table.core.windows.net/”

DataServiceContext context = new DataServiceContext(serviceUri);

TableStorageTable table = new TableStorageTable("Blogs");

// Tạo một bảng mới bằng cách thêm thực thể mới vào bảng chính gọi là "Tables"

context.AddObject("Tables", table);

// SaveChanges đưa đến một lời gọi đến máy chủ

DataServiceResponse response = context.SaveChanges();

## Thêm 1 Blog

Để thêm một thực thể, ứng dụng phải làm như sau.

1. Tạo một đối tượng C# mới và thiết lập tất cả các thuộc tính.
2. Tạo một đối tượng của DataServiceContext, đại diện cho một kết nối đến máy chủ trong dịch vụ lưu trữ ADO .NET cho tài khoản lưu trữ của bạn.
3. Thêm đối tượng C# vào context.
4. Gọi phương thức SaveChanges trên DataServiceContext để gửi yêu cầu đến máy chủ. Tại thời điểm này một truy vấn HTTP được gửi đến máy chủ với thực thể định dạng ATOM XML.

Sau đây là đoạn mã ví dụ :

Blog blog = new Blog {

PartitionKey = "Channel9", // ChannelName

RowKey = DateTime.UtcNow.ToString(), // PostedDate

Text = "Hello"**,**

Rating = 3

};

serviceUri = new Uri("http://<account>.table.core.windows.net");

var context = new DataServiceContext(serviceUri);

context.AddObject("Blogs", blog);

DataServiceContext response = context.SaveChanges();

## Truy vấn Blog

Thực thể được truy vấn sử dụng LINQ. Trong ví dụ này chúng ta lấy tất cả những blog có rating = 3.

Khi một truy vấn được liệt kê (ví dụ với lệnh foreach), một truy vấn được gửi đến máy chủ. Máy chủ gửi kết quả theo định dạng ATOM XML. Thư viện dịch vụ dữ liệu ADO .NET lưu kết quả vào đối tượng C# cho ứng dụng sử dụng.

var serviceUri = new Uri("http://<account>.table.core.windows.net");

DataServiceContext context = new DataServiceContext(serviceUri);

// Truy vấn LINQ sử dụng DataServiceContext cho tất cả các thực thể từ

// bảng Blogs có rating = 3

var blogs =

from blog in context.CreateQuery<Blog>("Blogs")

where blogs.Rating == 3

select blog;

foreach (Blog blog in blogs) { ... }

## Cập nhật một Blog

Để cập nhật một thực thể, ứng dụng nên làm như sau.

1. Tạo một DataContext với MergeOption được gán là OverwriteChanges hoặc PreserveChanges (sẽ được nói rõ ở mục ). Để chắc rằng Etag được đảm bảo một cách chính xác cho mỗi đối tượng mà nó nhận được.
2. Nạp thực thể cần cập nhật vào DataContext bằng LINQ. Lấy nó từ máy chủ sẽ đảm bảo rằng Etag đã được cập nhật trong thực thể và những thao tác cập nhật và xóa tiếp theo sẽ cập nhật Etag trong header if-match. Chỉnh sửa đối tượng C# đại diện cho thực thể.
3. Thêm đối tượng C# trở về cùng một DataContext để cập nhật. Sử dụng cùng DataContext đảm bảo chúng ta sẽ tự động tái sử dụng Etag đã được lấy trước đó cho đối tượng này.
4. Gọi SaveChanges để gửi truy vấn đến máy chủ.

Blog blog =

(from blog in context.CreateQuery<Blog>("Blogs")

where blog.PartitionKey == "Channel9"

&& blog.RowKey == "Oct-29"

select blog).FirstOrDefault();

blog.Text = "Hi there";

context.UpdateObject(blog);

DataServiceResponse response = context.SaveChanges();

## Xóa một Blog

Xóa một thực thể tương tự như cập nhật một thực thể. Sử dụng DataServiceContext để lấy thực thể và sau đó gọi phương thức DeleteObject trong context thay vì phương thức UpdateObject.

Blog blog =

(from blog in context.CreateQuery<Blog>("Blogs")

where blog.PartitionKey == "Channel9"

&& blog.RowKey == "Oct-29"

select blog).FirstOrDefault();

context.DeleteObject(blog);

DataServiceResponse response = context.SaveChanges();

## Giao dịch nhóm thực thể

Giao dịch nhóm thực thể hỗ trợ thực thi lên đến 100 lệnh thêm-xóa-sửa trong một lệnh batch đơn đối với thực thể trong một bảng và có cùng giá trị PartitionKey. Những lệnh này được thực hiện một thành công tất cả hoặc thất bại toàn bộ.

Ở đây có một vài thuật ngữ :

* Change set – Nhóm một hoặc nhiều lệnh Thêm-Xóa-Sửa cần được xử lí.
* Batch – Một batch chứa một hoặc nhiều change set hoặc truy vấn.

Sau đây là những giới hạn khi sử dụng lệnh batch cho Windows Azure Table:

* Chúng ta cho phép tối đa 100 thao tác trong một change set.
* Chúng ta cho phép chỉ một change set đơn hoặc truy vấn với mỗi thao tác batch. Truy vấn không thể được kết hợp thao tác Thêm-Xóa-Sửa trong một thao tác batch. Chúng ta chỉ hỗ trợ những truy vấn trả về chỉ một dòng, nghĩa là bộ lọc có PartitionKey và RowKey.
* Một batch đơn lẻ có thể nhiều nhất 4MB kích thước.
* Tất cả các lệnh trong change set phải thuộc cùng một phân vùng đơn lẻ trong một bảng duy nhất. Nghĩa là giá trị partition key phải giống nhau cho tất cả các thực thể và các thao tác nên xử lí dựa theo một bảng đơn lẻ.
* Mỗi thực thể có thể chỉ có xuất hiện một lần trong change set. Bạn không thể có change set mà thực thi nhiều thao tác trên cùng một thực thể. Change set có nghĩa thực hiện nhiều thao tác trên nhiều thực thể khác nhau trong cùng một phân vùng và bảng như một giao dịch nguyên tử.
* Một lệnh đơn lẻ trong change set được thực thi theo thứ tự mà chúng xuất hiện trong change set.

Trong khi đưa ra một giao dịch, ETag được gửi trong mỗi change set. Trên máy chủ, chúng tôi kiểm tra xem ETag có khớp với giá trị được lưu trữ trên máy chủ. Nếu có một Etag không khớp, toàn bộ giao dịch không thành công.

Sau đây tiếp tục ví dụ Blog cho thấy lợi ích của giao dịch nhóm thực thể để thêm, xóa, sửa qua nhiều thực thể từ một phân vùng trong một thao tác batch.

// newBlogs là danh sách những blog mới trong phân vùng Channel\_19

// deletedBlogs là danh sách những blog cần xóa trong phân vùng Channel\_19

// updatedBlogs is danh sách những blog cần cập nhật trong phân vùng Channel\_19

// Chúng ta giả định rằng deletedBlogs và updatedBlogs đã được nạp vào

// context trong những truy vấn trước đó.

// Thêm nhiều đối tượng blog cho phân vùng "Channel\_19".

// newBlogs là danh sách blog được chọn để thêm vào

for (int index = 0; index < newBlogs.Length; index++)

{

context.AddObject(newBlogs[index]);

}

// Xóa đối tượng blogs cho Channel\_19.

for (int index = 0; index < deletedBlogs.Length; index++)

{

context.DeleteObject(deletedBlogs[index]);

}

// Cập nhật đối tượng blogs ch Channel\_19.

// updatedBlogs là danh sách các Blog được chọn để tăng rating.

for (int index = 0; index < updatedBlogs.Length; index++)

{

updatedBlogs[index].Rating++;

context.UpdateObject(updatedBlogs[index]);

}

// Tất cả các thao tác Thêm/ Xóa/ Sửa được xử lí với một request batch

DataServiceResponse response =

context.SaveChanges(SaveChangesOptions.Batch);

Chỉ định SaveChangeOptions.Batch chỉ dẫn lệnh SaveChanges rằng tất cả những thay đổi chưa thực hiện được gom nhóm vào một change set đơn lẻ và gửi một giao dịch nhóm thực thể đơn lẻ đến hệ thống lưu trữ. Giao dịch batch sau đó được kiểm tra trên máy chủ để chắc rằng nó tuân thủ các hạn chế trên.

Bây giờ, nếu SaveChanges trên được thực hiện mà không có SaveChangesOptions.Batch, sau đó mỗi thực thể Thêm/Xóa/Sửa sẽ được gửi như một yêu cầu riêng lẻ đến hệ thống lưu trữ và thực thi riêng biệt.

Dưới đây là xử lí truy vấn đơn lẻ trong một lệnh batch. Chú ý, một lệnh batch có thể là một giao dịch nhóm thực thể như ở trên, hoặc truy vấn đơn lẻ như ở dưới.

// Thực hiện một truy vấn trong batch- Chú ý bộ lọc nên để theo dạng

// PartitionKey == 'Giá trị' && RowKey == 'Giá trị'

var q1 = from o in context.CreateQuery<Blogs>("Blogs")

where o.PartitionKey == "Channel\_19" &&

o.RowKey == "2"

select o;

DataServiceResponse response =

context.ExecuteBatch((DataServiceQuery<RetailStoreV1>)q1);

### Lỗi

Khi một truy vấn batch bị lỗi, sẽ không có thao tác nào thành công. Tuy nhiên, điều quan trọng là làm cho ứng dụng biết được cái lệnh gây ra toàn bộ giao dịch thất bại. Để xác định điều này, chúng ta trả về những lệnh được đánh số tuần tự trong thông báo lỗi như được thấy trong phần đánh dấu dưới đây. Tuy nhiên, nếu lỗi xảy ra bởi vì chứng thực hoặc bất kì lí do nào khác, điều này sẽ không được quy cho cho một lệnh đơn nào, do đó các con số sẽ không được cung cấp. Do đó, payload trong một sự kiện thất bại sẽ bao gồm:

1. Khóa của thao tác bị thất bại.
2. Các số không dựa trên chỉ số của hoạt động gây ra thất bại trong batch (chỉ số bắt đầu từ 0).
3. Chuỗi mô tả thất bại.

Ví dụ Response:

HTTP/1.1 202 Accepted

Cache-Control: no-cache

Transfer-Encoding: chunked

Content-Type: multipart/mixed; boundary=batchresponse\_7ab1553a-7dd6-44e7-8107-bf1ea1ab1876

Server: Table Service Version 1.0 Microsoft-HTTPAPI/2.0

x-ms-request-id: 45ac953e-a4a5-42ba-9b4d-97bf74a8a32e

Date: Thu, 30 Apr 2009 20:45:13 GMT

6E7

--batchresponse\_7ab1553a-7dd6-44e7-8107-bf1ea1ab1876

Content-Type: multipart/mixed; boundary=changesetresponse\_6cc856b4-8cb9-41eb-b8d2-bb73475c6cec

--changesetresponse\_6cc856b4-8cb9-41eb-b8d2-bb73475c6cec

Content-Type: application/http

Content-Transfer-Encoding: binary

HTTP/1.1 400 Bad Request

Content-ID: 4

Content-Type: application/xml

Cache-Control: no-cache

DataServiceVersion: 1.0;

<?xml version="1.0" encoding="utf-8" standalone="yes"?>

<error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">

<code>InvalidInput</code>

<message xml:lang="en-US">1:One of the request inputs is not valid.</message>

</error>

--changesetresponse\_6cc856b4-8cb9-41eb-b8d2-bb73475c6cec--

--batchresponse\_7ab1553a-7dd6-44e7-8107-bf1ea1ab1876--

0

Khi một lỗi xảy ra, bạn có thể lấy tuần tự số lệnh bằng đoạn mã sau :

try

{

// ... save changes

}

catch (InvalidOperationException e)

{

DataServiceClientException dsce = e.InnerException as DataServiceClientException;

int? commandIndex;

string errorMessage;

ParseErrorDetails(dsce, out commandIndex, out errorMessage);

}

void ParseErrorDetails(

DataServiceClientException e,

out string errorCode,

out int? commandIndex,

out string errorMessage)

{

GetErrorInformation(e.Message, out errorCode, out errorMessage);

commandIndex = null;

int indexOfSeparator = errorMessage.IndexOf(':');

if (indexOfSeparator > 0)

{

int temp;

if (Int32.TryParse(errorMessage.Substring(0, indexOfSeparator), out temp))

{

commandIndex = temp;

errorMessage = errorMessage.Substring(indexOfSeparator + 1);

}

}

}

void GetErrorInformation(

string xmlErrorMessage,

out string errorCode,

out string message)

{

message = null;

errorCode = null;

XName xnErrorCode = XName.Get(

"code",

"http://schemas.microsoft.com/ado/2007/08/dataservices/metadata");

XName xnMessage = XName.Get(

"message",

"http://schemas.microsoft.com/ado/2007/08/dataservices/metadata");

using (StringReader reader = new StringReader(xmlErrorMessage))

{

XDocument xDocument = null;

try

{

xDocument = XDocument.Load(reader);

}

catch (XmlException)

{

// The XML could not be parsed. This could happen either because the connection

// could not be made to the server, or if the response did not contain the

// error details (for example, if the response status code was neither a failure

// nor a success, but a 3XX code such as NotModified.

return;

}

XElement errorCodeElement =

xDocument.Descendants(xnErrorCode).FirstOrDefault();

if (errorCodeElement == null)

{

return;

}

errorCode = errorCodeElement.Value;

XElement messageElement =

xDocument.Descendants(xnMessage).FirstOrDefault();

if (messageElement != null)

{

message = messageElement.Value;

}

}

}

# Cập nhật đồng thời

Để cập nhật một thực thể yêu cầu những bước sau.

1. Lấy thực thể từ máy chủ.
2. Cập nhật đối tượng cục bộ, và gửi nó trở lại máy chủ.

Chúng ta giả định rằng có hai xử lí đồng thời cố gắng cập nhật cùng một thực thể. Bởi vì bước 1 và bước 2 không phải là atomic,nên một trong số chúng có thể kết thúc thay đổi phiên bản cũ của thực thể. Windows Azure Table sử dụng mô hình optimistic concurrency để giải quyết vấn đề này.

1. Mỗi thực thể có một phiên bản được duy trì trên hệ thống. Phiên bản này được thay đổi bởi máy chủ sau mỗi lẫn cập nhật.
2. Khi một thực thể được lấy ra, máy chủ gửi phiên bản này về phía client bằng HTTP ETag.
3. Khi client gửi yêu cầu cập nhật đến máy chủ, nó gửi Etag này đến máy chủ bằng header If-Match.
4. Nếu phiên bản của thực thể trên máy chủ giống với Etag trong header If-Match, việc thay đổi được chấp nhận, và thực thể được lưu trữ lấy phiên bản mới nhất trên máy chủ. Phiên bản mới được trả về phía client bằng Etag header.
5. Nếu phiên bản của thực thể trên máy chủ khác với Etag trong header If-Match, việc thay đổi bị từ chối, và lỗi HTTP “điều kiện tiên quyết thất bại” được trả về phía client.

Khi ứng dụng client nhận được lỗi “điều kiện tiên quyết không thành công”, hành vi đặc trưng của nó là thử lại thao tác bằng đoạn mã phía bên dưới.

1. Nó nên thử lại đối tượng một lần nữa, do đó lấy được phiên bản mới nhất.
2. Thực hiện cập nhật cục bộ và gửi nó trở lại máy chủ.

Nếu ứng dụng sử dụng thư viện .NET client, mã lỗi HTTP được biểu hiện đến ứng dụng qua exception (DataServiceRequestException).

Trong ví dụ bên dưới, hai client khác nhau thực thi cùng đoạn mã để thay đổi nội dung. Hai client gán thuộc tính “Text” với hai giá trị khác nhau. Sau đây là một chuỗi các sự kiện có khả năng minh họa làm thế nào để cập nhật đông thời được xử lí.

1. Cả hai client lấy thực thể về. Và cũng lấy Etag theo thực thể ví dụ là “v1”.
2. Mỗi client cập nhật thuộc tính Text ở cục bộ.
3. Mỗi client gọi UpdateObject và SaveChanges.
4. Mỗi client gửi truy vấn HTTP đến máy chủ với header “If-Match:v1”
5. Một trong những client đến máy chủ trước.
   1. Máy chủ so sánh header “If-Match” với phiên bản của thực thể. Chúng bằng nhau
   2. Máy chủ thực hiện thay đổi.
   3. Thực thể nhận một phiên bản mới “v2” trên máy chủ.
   4. Một header mới “Etag:v2” được gửi lại qua response đến client.
6. Client khác đến máy chủ tiếp theo. Tại thời điểm này, client đầu tiên đã thay đổi ứng dụng.
   1. Máy chủ so sánh header “If-Match” với phiên bản của thực thể. Chúng không bằng nhau, bởi vì phiên bản của thực thể đã được thay đổi thành “v2”, trong khi phiên bản của request là “v1”.
   2. Máy chủ từ chối truy vấn.

context.MergeOption = MergeOption.PreserveChanges;

Blog blog =

(from blog in context.CreateQuery<Blog>("Blogs")

where blog.PartitionKey == "Channel9"

&& blog.RowKey == "Oct-29"

select blog).FirstOrDefault();

blog.Text = "Hi there again";

try

{

context.UpdateObject(blog);

DataServiceResponse response = context.SaveChanges();

}

catch (DataServiceRequestException e)

{

OperationResponse response = e.Response.First();

if (response.StatusCode == (int)HttpStatusCode.PreconditionFailed)

{

// query the object again to retrieve the latest etag

// and update it

}

}

## Cập nhật không điều kiện

Nếu một ứng dụng muốn cập nhật một thực thể không điều kiện nên thực hiện theo những bước sau

1. Tạo mới một DataServiceContext hoặc nếu sử dụng một DataServiceContext tồn tại, tách rời đối tượng như ví dụ bên dưới.
2. Nối thực thể với context và sử dụng “\*” bằng giá trị Etag mới.
3. Cập nhật thực thể.
4. Gọi SaveChanges.

// gán merge option thành overwrite để cho phép các thực thể được cập nhật

context.Detach(blog);

context.AttachTo("Blogs", blog, "\*");

blog.Text = "Hi there again";

try

{

context.UpdateObject(blog);

DataServiceResponse response = context.SaveChanges();

}

catch (DataServiceRequestException e)

{

// Lỗi xử lí – nhưng có không quăng lỗi PreCondition

}

# Phân trang các kết quả truy vấn

Đối với các truy vấn có thể trả lại số lượng lớn các kết quả, hệ thống cung cấp hai cơ chế:

1. Có khả năng lấy N thực thể đầu tiên sử dụng chức năng Take(N) của LINQ.
2. Một thẻ tiếp nối cho biết nơi tập các kết quả tiếp theo bắt đầu.

## Lấy N thực thể trên cùng

Hệ thống hỗ trợ trả về N thực thể trên cùng phù hợp với truy vấn. Ví dụ, trong .Net, bạn có thể sử dụng chức năng Take(N) của LINQ để lấy N thực thể trên cùng.

Lấy 100 thực thể trong ví dụ dưới đây.

serviceUri = new Uri("http://<account>.table.core.windows.net");

DataServiceContext svc = new DataServiceContext(serviceUri);

var allBlogs = context.CreateQuery<Blog>("Blogs");

foreach (Blog blog in allBlogs.Take(100))

{

// do something with each blog

}

Chức năng này cũng được hỗ trợ qua giao thức REST với chuỗi truy vấn $top=N. Ví dụ, truy vấn “GET http://<serviceUri>/Blogs?$top=100” sẽ được trả về 100 thực thể trên cùng, ứng với truy vấn. Việc lọc được thực hiện trên máy chủ lấy chỉ 100 thực thể để gửi trong response trả về cho client.

## Thẻ tiếp nối

Số lượng thực thể được trả về trong truy vấn có thể bị giới hạn bởi một trong những nguyên nhân sau :

1. Truy vấn xác định số lượng tối đa thực thể được trả về.
2. Số lượng thực thể lớn hơn số lượng thực thể tối đa được cho phép trong một response từ máy chủ (hiện tại là 1000).
3. Số lượng kích thước thực thể trong một response lớn hơn kích thước tối đã của một response (hiện tại là 4MB bao gồm tên thuộc tính nhưng loại trừ khi cú pháp xml được sử dụng cho REST).
4. Thời gian xử lí lâu hơn thời gian chờ của máy chủ (hiện tại là 60 giây).

Trong mỗi trường hợp này, response bao gồm một thẻ nối tiếp qua header custom. Đối với một truy vấn trên các thực thể của bạn, header custom đại hiện một thẻ nối tiếp là :

* x-ms-continuation-NextPartitionKey
* x-ms-continuation-NextRowKey

Truy vấn tiếp theo sẽ như sau

http://<serviceUri>/Blogs?<originalQuery>&NextPartitonKey=<someValue>&NextRowKey=<someOtherValue>

Client sẽ lặp lại điều này cho đến khi không còn thẻ tiếp nối. Lúc này, tất cả những kết quả phù hợp đã được lấy ra.

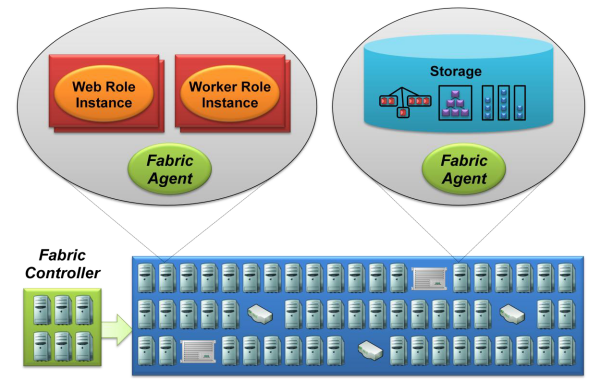
Thẻ tiếp nối phải được xem như một giá trị mờ. Nó phản ảnh thời điểm bắt đầu cho truy vấn tiếp theo, và nó có thể không tương ứng với thực thể thực tế trong bảng. Nếu một thực thể mới được thêm vào có Khóa(thực thể mới) > Khóa (thực thể cuối cùng nhận được trong truy vấn), nhưng Khóa(thực thể mới) < “Thẻ tiếp tục”, thì thực thể mới này sẽ không được trả về khi truy vấn thực hiện lại sử dụng thẻ tiếp tục. Tuy nhiên, nếu thực thể mới được thêm vào có khóa (thực thể mới) > thẻ tiếp tục, nó sẽ được trả về trong tập con truy vấn sử dụng thẻ tiếp tục.

#### 3.1.4.3. SQL Azure

( Bổ sung sau).

#### 3.1.4.3. Fabric

Tất cả các ứng dụng Windows Azure, và dữ liệu của nó đều tồn tại trên trung tâm dữ liệu của Microsoft. Bên trong trung tâm dữ liệu này, một tập hợp các máy dành cho Windows Azure được tổ chức thành một kết cấu (*fabric*).



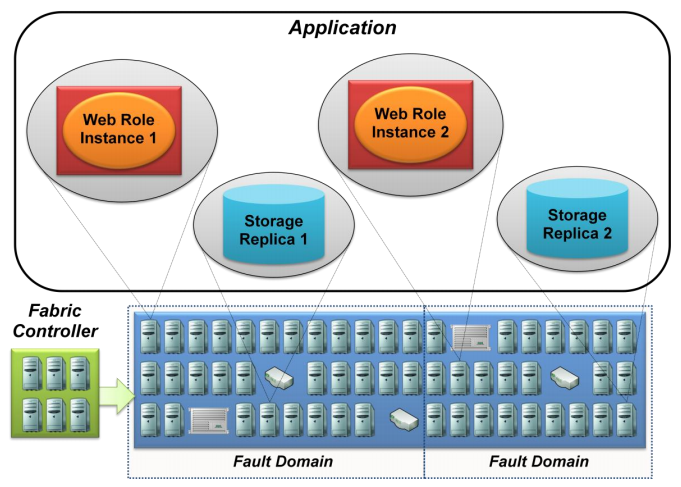
Hình: Fabric Controller

Như hình trên, Windows Azure Fabric chứa một một nhóm các máy, tất cả chúng được quản lý bởi một phần mềm gọi là *fabric controller*. Fabric controller được tái tạo qua mỗi nhóm từ 5 đến 7 máy, nó là sở hữu tất cả các tài nguyên: máy tính, switches, bộ cân bằng tải (load balancer),… Bởi vì nó có thể giao tiếp với một *fabric agent* trên mỗi máy tính, nó cũng nhận biết được tất cả các ứng dụng Windows Azure trong kết cấu.

Với các thông tin có được, cho phép *fabric controller* có thể làm được nhiều việc rất hữu ích. Nó theo dõi tất cả các ứng dụng đang chạy. Nó quản lý hệ điều hành, quản lý các việc như vá lỗi cho phiên bản của Windows Service 2008. Nó quyết định khi một ứng dụng mới được upload lên, thì sẽ được chạy trên dịch vụ nào. Để làm được điều này, fabric controller phụ thuộc vào tập tin cấu hình của mỗi ứng dụng được upload lên, trong đó, chỉ ra bao nhiêu thể hiện cần được tạo ra, và kích thước các máy ảo là thế nào. Dựa vào đó, fabric controller tạo ra các máy ảo tương ứng. Khi tạo ra các máy ảo này, fabric controller sẽ theo dõi các ứng dụng đó, Nếu một, ứng dụng, cần có 5 thể hiện, và một trong số đó bị “chết”, fabric controller sẽ tự động khởi tạo một thể hiện mới. Nếu một máy ảo đang chạy bị chết, fabric controller sẽ tự động khởi tạo một thể hiện khác của ứng dụng trên một máy ảo khác, sau đó khởi động lại bộ cân bằng tải nếu cần thiết để chỉ đến máy mới này.

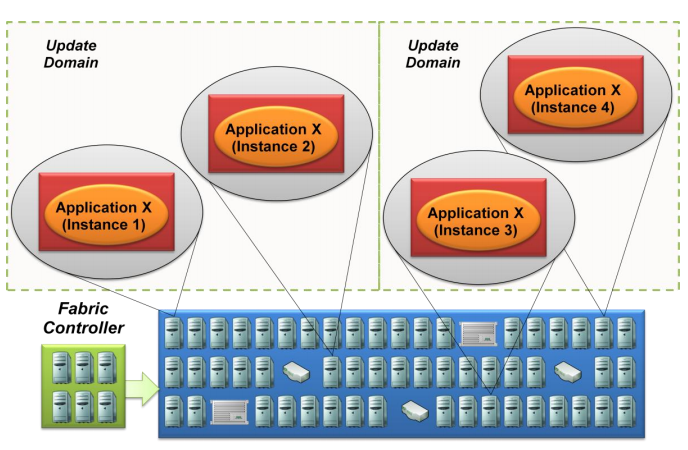
Như đã nói, fabric controller chịu trách nhiệm gán cho các thể hiện của ứng dụng vào một máy vật lý cụ thể. Điều này, rất quan trọng, việc gán thế nào, sẻ ảnh hướng rất nhiều đến tính sẵn sàng phục vụ của ứng dụng. Ví dụ, một ứng dụng yêu cầu 5 thể hiện Web role, 2 thể hiện Worker role, giả sử fabric controller gán các thể hiện này vào một mạng dùng chung một switch, vậy điều gì xẩy ra, nếu như switch này không hoạt động được, ứng dụng sẽ không còn hoạt động được nữa. Đích hướng dến của Windows Azure là tính sẵn sàng cao, do vây, việc để cho ứng dụng phụ thuộc vào một điểm thất bại là không chấp nhận được.

Để khắc phục điều này, fabric controller nhóm các máy nó làm chủ thành một số các miền gọi là *fault domains*. Mỗi miền là một phần của trung tâm dữ liệu, Ví dụ :



Trong hình trên, giả sử ứng dụng cần 2 thể hiện Web role, và trung tâm dữ liệu được chia làm 2 miền lỗi. Khi fabric controller triển khai ứng dụng này, nó sẽ đặt mỗi thể hiện Web role vào một miền, khi đó, khi có lỗi xẩy ra ở một miền nào đó, cũng sẽ không thể làm chết hoàn toàn ứng dụng của bạn.

Điều này vẫn chưa đủ, điều gì xảy ra nếu ứng dụng có nhu cầu cập nhật, việc tắt ứng dụng và khở động lại là không nên. Để tránh điều này, fabric controller nhóm các thể hiện của ứng dụng thành các miền cập nhât “*update domain*”.



Khi mã của ứng dụng cần cập nhât, fabric controller sẽ thực hiện cập nhật trong từng miền. Ví dụ trong hình trên, fabric controller sẽ tiến hành tắt 2 thể hiện 1 và 2 của ứng dụng trước, cập nhật,và khởi động lại. Sau đó, tiến hành tương tự cho 2 thể hiện ở miền còn lại. Mục đích của việc này là làm cho ứng dụng không bi gián đoạn, khi một ứng dụng đang cập nhật, người dùng vẫn có thể truy cập đến với phiên bản cũ của ứng dụng, cho dến khi nó cập nhật xong.

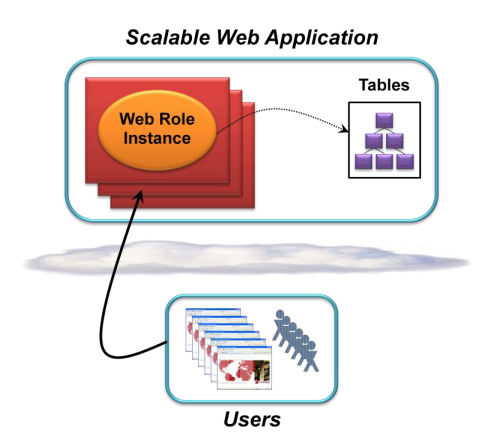
#### 3.1.4.4. Các kịch bản sử dụng Windows Azure

Chúng ta sẽ đi qua 5 kịch bản chính sử dụng Windows Azure: tạo một ứng dụng web có khả năng mở rộng, tạo một ứng dụng xử lí song song, tạo một ứng dụng web với xử lí nền, tạo một ứng dụng web với dữ liệu quan hệ, và một ứng dụng on-premise hoặc hosted sử dụng bộ lưu trữ đám mây.

##### **Tạo môt ứng dụng web có khả năng mở rộng**

Ví dụ, nếu ứng dụng có nhu cầu xử lí nhiều người dùng truy xuất đồng thời, thì cần phải xây dựng nó trên một nền tảng hỗ trợ mở rộng ứng dụng và mở rộng dữ liệu. Hoặc ứng dụng có tải thay đổi đáng kể, thỉnh thoảng có nhu cầu sử dụng tải cực cao. Ví dụ như: bán vé online hoặc xem các tin tức nóng hổi có thể thể hiện mô hình này. Chạy ứng dụng loại này trong một trung tâm dữ liệu thông thường đòi hỏi phải luôn có sẵn đủ máy để xử lý các tải ở đỉnh cao, mặc dù toàn bộ hệ thống không sử dụng hầu hết thời gian. Nếu ứng dụng được xây dựng trên Windows Azure, tổ chức chạy nó có thể mở rộng nhiều thể hiện để sử dụng khi cần thiết, sau đó co lại với số thể hiện ít hơn. Bởi vì Windows Azure tính phí theo nhu cầu sử dụng – điều này sẽ rẻ hơn việc duy trì nhiều máy không sử dụng tới.

Để tạo một ứng dụng web có thể mở rộng trên Windows Azure, lập trình viên có thể sử dụng Web role và table.

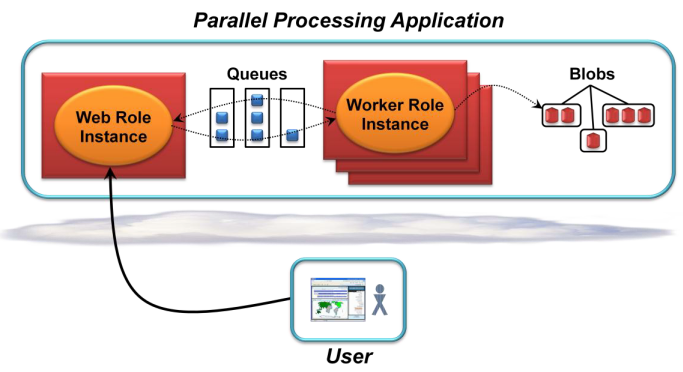


**Hình Ứng dụng Web mở rộng sử dụng Web Role và table**

Trong trường hợp này, lập trình viên xác định số thể hiện mà ứng dụng nên chạy, và Windows Azure fabric controller tạo ra số máy ảo tương ứng. Như đã nói trước đó, fabric controller cũng theo dõi các thể hiện, và đảm bảo rằng số lượng thể hiện yêu cầu là luôn luôn có sẵn. Để lưu trữ dữ liệu, ứng dụng sử dụng Windows Azure Table, nó cung cấp khả năng lưu trữ mở rộng để xử lý một lượng lớn dữ liệu.

##### **Tạo môt ứng dụng xử lí song song**

Lập trình viên có thể sử dụng nhiều Worker role để tạo ra ứng dụng loại này. Và nó không chỉ là lựa chọn duy nhất, xử lí song song có thể sử dụng một tập lớn dữ liệu,nó có thể được lưu trữ trong Window Azure Blob.



Hình 7 Ứng dụng xử lí song song dùng 1 Web role, nhiều Work role , blob và queue

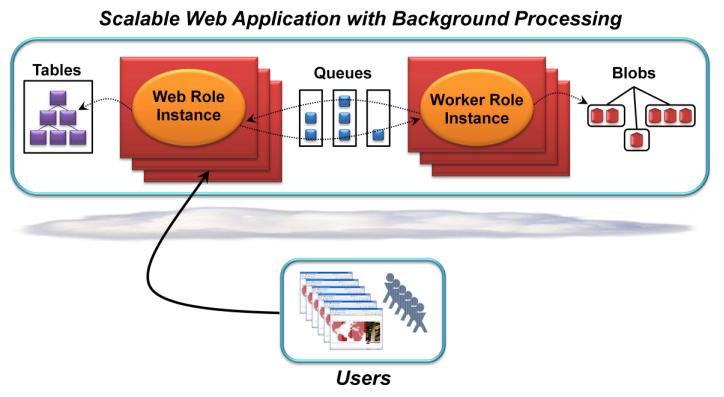
Trong chiến lược ở đây, công việc song song được thực hiện bằng nhiều thể hiện Worker role chạy đồng thời, mỗi thể hiện sử dụng dữ liệu blob. Để tương tác với ứng dụng người dùng dựa vào một thể hiện Web role. Qua giao tiếp này, người dùng có thể quyết định bao nhiêu thể hiện Worker role nên chạy, bắt đầu và dừng lại các thể hiện, lấy kết quả,…Liên lạc giữa thể hiện Web role và Worker role dựa trên Windows Azure Storage queue.

##### **Tạo một ứng dụng Web mở rộng với xử lí nền**

Trong nhiều tình huống phần mềm truy xuất trình duyệt cũng cần thiết lập các công việc chạy nền độc lập với request/ response của ứng dụng.

Ví dụ, ứng dụng web chia sẻ video. Nó cần phải chấp nhận request từ số lượng lớn người dùng đồng thời. Một số request sẽ tải lên video mới, mỗi video phải được xử lý và lưu trữ cho truy cập sau đó. Làm cho người dùng chờ đợi trong khi xử lý này đang thực hiện sẽ làm mất nhiều thời gian. Thay vì vậy, một phần của ứng dụng chấp nhận request có thể thiết lập một công việc nền để thực hiện công việc này.

Nhiều Windows Azure Web role và Worker role được sử dụng cùng nhau trên kịch bản này.

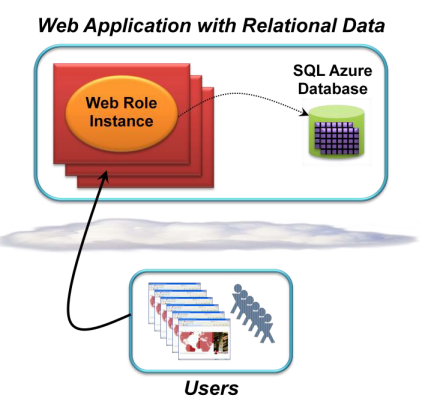


Hình 8 Một ứng dụng Web mở rộng với xử lí nền

Ứng dụng xử dụng nhiều thể hiện Web role để xử lí các request từ người dùng, Để hỗ trợ số lượng lớn các người dùng đồng thời, table được sử dụng để lưu trữ thông tin. Xử lí nền, nó dựa vào các thể hiện Worker role, chuyển các công việc qua queue.

##### **Tạo một ứng dụng Web với dữ liệu quan hệ**

Blob, table, và queue phù hợp với một số tình huống. Trong những tình huống khác, dữ liệu quan hệ phù hợp hơn. Giả sử một doanh nghiệp muốn chạy một ứng dụng trên Windows Azure. Ứng dụng này không cần thiết phải mở rộng dữ liệu hay thay đổi thành phần dữ liệu như Windows Azure Table hỗ trợ. Thay vào đó, lập trình viên thích sử dụng dữ liệu quan hệ hơn. Trong trường hợp này, ứng dụng có thể sử dụng Windows Azure cùng với cơ sở dữ liệu SQL Azure.

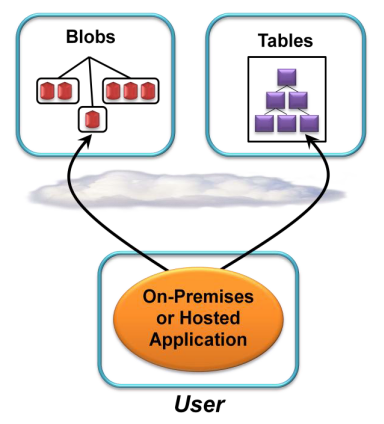


Hình 9 Ứng dụng Windows Azure sử dụng cơ sở dữ liệu SQL Azure.

Cơ sở dữ liệu SQL Azure cung cấp một tập lớn các chức năng SQL Server khi quản lí dịch vụ đám mây. Ứng dụng có thể tạo cơ sở dữ liệu, chạy truy vấn SQL,và hơn thế nữa, nhưng không cần quản lí hệ thống cơ sở dữ liệu hoặc phần cứng – Microsoft sẽ lo việc này. Bởi vì cơ sở dữ liệu SQL Azure là dịch vụ đám mây, việc tính phí dựa trên lưu lượng sử dụng.

##### **Sử dụng lưu trữ đám mây từ ứng dụng on-premise hoặc hosted.**

Trong khi Windows Azure cung cấp nhiều khả năng, một ứng dụng đôi khi chỉ cần sử dụng một trong số đó. Ví dụ, hãy nghĩ về một ứng dụng on-premises và hosted có nhu cầu để lưu trữ lượng dữ liệu rất lớn. Một trang web tin tức chạy ở một hoster có thể cần một nơi có khả năng mở rộng và truy xuất trên toàn cầu để lưu trữ số lượng lớn văn bản, đồ họa, video, và thông tin hồ sơ về người sử dụng.



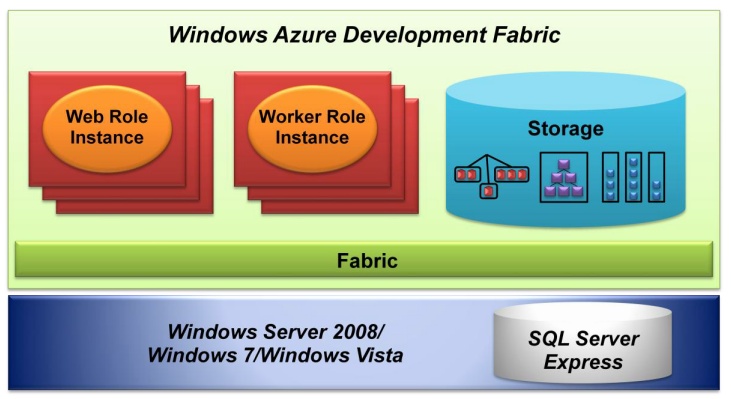
**Hình 10: Một ứng dụng on-premises hoặc hosted dùng Windows Azure blob và table.**

Như hình trên, một ứng dụng on-premise hoặc hosted có thể truy xuất trực tiếp bộ lưu trữ Windows Azure. Với truy cập này có thể sẽ chậm hơn so với truy cập lưu trữ cục bộ, nhưng nó cũng có thể sẽ rẻ hơn, và có khả năng mở rộng lớn hơn. Đối với một số ứng dụng cần phải cân nhắc vấn đề này. Ngoài ra, ứng dụng có thể sử dụng cơ sở dữ liệu SQL Azure.

##### **Phát triển ứng dụng Windows Azure**

Đối với các lập trình viên, xây dựng ứng dụng Windows Azure cũng giống như xây dựng một ứng dụng Windows truyền thống. Nền tảng Windows Azure hỗ trợ cả ứng dụng .Net và ứng dụng xây dựng với native code. Windows Azure cung cấp các mẫu project trong Visual Studio để tạo Web role, Worker role, và kết hợp cả hai.

Một vấn đề là ứng dụng Windows Azure không chạy cục bộ mà chạy trên đám mây. Vì vậy, để giải quyết vấn đề này, Microsoft cung cấp development fabric, một phiên bản của môi trường Windows Azure để chạy trên máy của lập trình viên.



Hình 11 Development fabric cung cấp một bản sao của Windows Azure cho lập trình viên.

Development fabric chạy trên máy đơn Windows Server 2008, Windows 7, hoặc Windows Vista. Nó mô phỏng các chức năng của Windows Azure trên đám mây, với Web role, Worker role, và bộ lưu trữ Windows Azure. Lập trình viên có thể xây dựng một ứng dụng Windows Azure triển khai nó trên development fabric, và chạy nó giống như chạy ứng dụng trên đám mây. Lập trình viên có thể quyết định xem bao nhiêu thể hiện của mỗi role nên chạy và có thể dùng queue để liên lạc giữa các thể hiện với nhau. Một ứng dụng có thể được phát triển và kiểm thử ở cục bộ, lập trình viên có thể upload code và tập tin cấu hình của nó qua cổng Windows Azure, sau đó chạy nó.

## 3.2. Ứng dụng trên điện toán đám mây

Ứng dụng sử dụng điện toán đám mây dễ nhận ra nhất chính là sản phẩm của Google mà chúng ta sử dụng hằng ngày: ChromeBook. Bạn chỉ cần đăng nhập thông tin của mình bằng gmail ở bất cứ đâu, sau 1 khoảng thời gian thiết lập, chrome sẽ nhận ra các thoi quen sử dụng trình duyệt của chủ nhân (hình nền, các bookmark,... ).

Bên cạnh Email thì mạng xã hội cũng là một lĩnh vực mà cloud computing được ứng dụng khá rộng rãi. Dễ thấy nhất là Facebook, MySpace. Ý tưởng chính của mạng xã hội là để tìm người bạn đã biết hoặc những người bạn sẽ muốn biết và chia sẻ thông tin của bạn với họ. Tất nhiên, khi bạn chia sẻ thông tin của bạn với những người này, bạn cũng chia sẻ nó với những người chạy dịch vụ.

Những sản phẩm của [Google Docs](http://translate.googleusercontent.com/translate_c?hl=vi&ie=UTF8&prev=_t&rurl=translate.google.com.vn&sl=en&tl=vi&twu=1&u=http://www.google.com/google-d-s/intl/en/tour1.html&usg=ALkJrhhTuMV2aE--cMYnDVMqChpNIFgyWw) ,cũng như một số dịch vụ [Zoho Office](http://translate.googleusercontent.com/translate_c?hl=vi&ie=UTF8&prev=_t&rurl=translate.google.com.vn&sl=en&tl=vi&twu=1&u=http://www.zoho.com/&usg=ALkJrhg8FnV6Vary8OQICe23qj1ApwDPHg) tồn tại trên Internet, nó cho phép bạn giữ và chỉnh sửa tài liệu của bạn trực tuyến . Bằng cách đó, các tài liệu sẽ có thể truy cập bất cứ nơi nào, và bạn có thể chia sẻ tài liệu và tương tác với chúng. Nhiều người có thể làm việc trong cùng một tài liệu cùng một lúc.

Google và Microsoft cung cấp một phương tiện mà người tiêu dùng có thể tạo ra một hồ sơ sức khỏe trực tuyến cá nhân (PHR [)](https://www.google.com/health) . Google và Microsoft HealthVault cho phép người dùng tạo ra, lưu trữ, và truy cập hồ sơ sức khỏe trực tuyến cá nhân trên website tìm kiếm cả mình.

Ngay cả khi bạn sử dụng dịch vụ để giữ tất cả các tài liệu và hình ảnh, rất có thể là bạn vẫn còn có dữ liệu trên máy tính cá nhân của bạn. Một trong những vấn đề lớn nhất với máy tính cá nhân có xu hướng bị mất dữ liệu đó nếu máy tính của bạn bị đánh cắp, bị phá hủy, hoặc thiết bị lưu trữ bị hư hỏng. Các dịch vụ như [Syncplicity](http://translate.googleusercontent.com/translate_c?hl=vi&ie=UTF8&prev=_t&rurl=translate.google.com.vn&sl=en&tl=vi&twu=1&u=http://www.syncplicity.com/&usg=ALkJrhi4cuOe_GbWs_Hfi9nz9LuYRpXWVg) và [Dropbox](http://translate.googleusercontent.com/translate_c?hl=vi&ie=UTF8&prev=_t&rurl=translate.google.com.vn&sl=en&tl=vi&twu=1&u=http://www.dropbox.com/&usg=ALkJrhimDPI1_aVwpsPmNumamp-l74E1vg) (cả hai phiên bản cung cấp miễn phí) làm cho nó dễ dàng để giữ các bản sao của các tập tin trên nhiều máy tính đồng bộ trong khi vẫn giữ một bản sao trong " đám mây ". Một số các dịch vụ này thậm chí sẽ tiếp tục các phiên bản trước của tập tin hoặc file bị xóa trong trường hợp bạn xóa một file quan trọng.

## 3.3. Đề tài sổ liên lạc trực tuyến

### 3.3.1. Giới thiệu tổng quan

Hệ thống sổ liên lạc trực tuyến dành cho các trường Tiểu học, THCS, PTTH góp phần tăng cường giảng dạy, đào tạo, đẩy mạnh ứng dụng Công nghệ thông tin trong ngành giáo dục, xây dựng nhà trường thân thiện, học sinh tích cực đúng như mong muốn và mục tiêu của Bộ GD&ĐT đã đề ra.

Hệ thống sổ liên lạc trực tuyến được nhóm xây dựng trên mô hình là kênh thông tin giữa nhà trường và phụ huynh qua Internet bằng ứng dụng web và email. Qua kênh này, phụ huynh nhanh chóng nhận được:

* Thông tin về lớp học
* Thông tin về điểm của con em mình
* Nhận xét của giáo viên
* Các thông báo của nhà trường như họp phụ huynh, nghỉ lễ, tết

Đồng thời, phụ huynh cũng có thể gửi ý kiến phản hồi tới nhà trường, giáo viên.

Với việc áp dụng hệ thống Sổ liên lạc trực tuyến này sẽ đem lại nhiều lợi ích cho Nhà trường và Phụ huynh học sinh như:

* Đối với Phụ huynh học sinh:
  + Dễ dàng nắm bắt thông tin về tình hình học tập hàng ngày của con em cũng như các nhận xét, thông báo, mời họp từ phía nhà trường, kế hoạch của lớp.
* Đối với nhà trường:
  + Nâng cao chất lượng giảng dạy do phụ huynh theo dõi sát sao tới con em mình hơn
  + Nâng cao hình ảnh về một nhà trường hiện đại, chuyên nghiệp
  + Góp phần thúc đẩy hiện đại hóa, tin học hóa trong nhà trường

Ngoài ra Sổ liên lạc trực tuyến còn mang tính bảo mật trong việc cung cấp thông tin. Hệ thống sẽ kiểm chứng tin nhắn gửi đến đúng số ĐTDĐ đã đăng ký, để đảm bảo không ai khác ngoài phụ huynh có thể nhận thông tin của con em mình.

Với những lợi ích trên, cùng với các chức năng giải pháp mềm dẻo, linh hoạt và hệ thống đã được trải nghiệm tại nhiều trường, nhóm chúng tôi tin tưởng khi hệ thống đưa vào sử dụng sẽ làm nâng cao được chất lượng giảng dạy và làm hài lòng Nhà trường, phụ huynh.

Hệ thống sổ liên lạc trực tuyến là công cụ giúp phụ huynh, giáo viên, các đơn vị quản lý nắm bắt được thông tin học tập, đạo đức của con em mình tại trường nhanh chóng – chính xác - kịp thời

### 3.3.2. Lí do chọn đề tài sổ liên lạc trực tuyến

Qua nghiên cứu các lợi ích mà điện toán đám mây mang đến, nhóm chúng tôi chọn xây dựng hệ thống sổ liên lạc trực tuyến bởi vì điện toán đám mây mang đến các giải pháp cho hệ thống sổ liên lạc trực tuyến như sau:

* **Giải quyết được bài toán đa người dùng**

Hệ thống sổ liên lạc trực tuyến được xây dựng để áp dụng cho nhiều trường học, do đó lượng người sử dụng hệ thống là rất lớn. Với sự hỗ trợ về khả năng lưu trữ, điện toán đám mây là giải pháp tốt cho bài toán này.

* **Chi phí sử dụng**

Sổ liên lạc trực tuyến tuy không phải là hình thức liên lạc mới nhưng tính chất phổ biến ở các trường học chưa cao. Việc tính toán, ước lượng lượng người sử dụng hệ thống ở mỗi trường không đồng nhất và gặp nhiều khó khăn. Đế áp dụng hình thức liên lạc trực tuyến này, không phải tất cả các trường đều có khả năng kinh tế để xây dựng hệ thống cho riêng mình hoặc không phải tất cả nhà cung cấp dịch vụ có khả năng xây dựng hệ thống dành cho các trường theo mô hình client/server truyền thống. Do đó, với giải pháp tính toán chi phí theo lưu lượng sử dụng, điện toán đám mây là lựa chọn tối ưu để xây dựng hệ thống sổ liên lạc trực tuyến.

* **Chi phí bảo trì**

Nếu xây dựng theo mô hình client/server truyền thống, nhà trường phải tự mình hoặc nhà cung cấp dịch vụ phải tự quản lí hạ tầng của hệ thống, giải quyết các vấn đề liên quan đến mạng, v.v…Nhưng với điện toán đám mây, phía trường hoặc nhà cung cấp dịch vụ chỉ cần quan tâm đến nghiệp vụ của sổ liên lạc. Bởi vì điện toán đám mây sẽ tự giải quyết các vấn để liên quan đến hạ tầng.

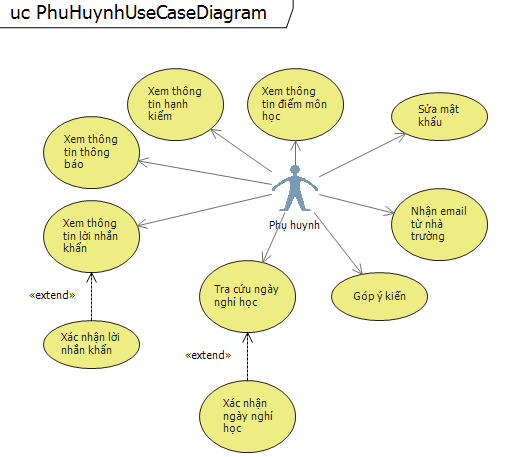
### 3.3.3. Tính năng chính

Các tính năng chính của hệ thống sổ liên lạc trực tuyến:

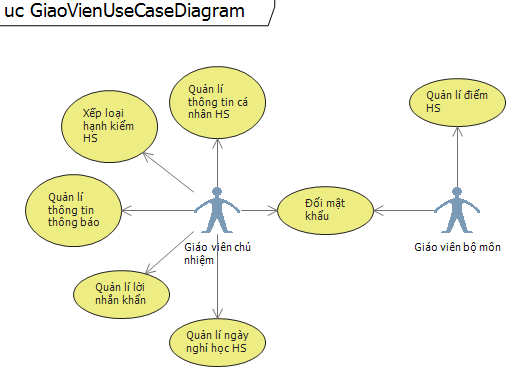
* Đối với nhà trường:
  + Góp phần phục vụ nghị định chung của Chính phủ
  + Quản lý học sinh bằng CNTT hiệu quả
  + Môi trường học tập thân thiện
* Đối với phụ huynh:
  + Kiểm tra điểm môn học
  + Kiểm tra mức độ chuyên cần, ý thức đạo đức
  + Xem thông báo bất thường của nhà trường
  + Nhận các tư vấn về giáo dục của chính giáo viên nhà trường
  + Được cấp Account gồm Username, và Password để đăng nhập vào hệ thống bất cứ lúc nào
  + Đóng góp các ý kiến, thông tin và phản hồi về nhà trường
* Đối với cán bộ IT:
  + Công cụ phần mềm quản lý đơn giản, hiệu quả, không tốn thời gian
  + Là nguồn tài liệu lưu trữ thông tin học sinh, lớp, khối
* Đối với Giáo viên:
  + Quản lý học sinh bất cứ thời điểm nào
  + Quản lý chung lớp học khi không có mặt tại trường
  + Được quản lý tổng thể từng cá nhân, từng lớp, khối theo tiêu chí từng tuần, từng tháng, từng học kỳ.
  + So sánh được tổng quan về tình hình học tập, đạo đức của lớp
  + Liên lạc được ngay với phụ huynh khi có thông tin đặc biệt

## 3.4. Phát triển bài toán

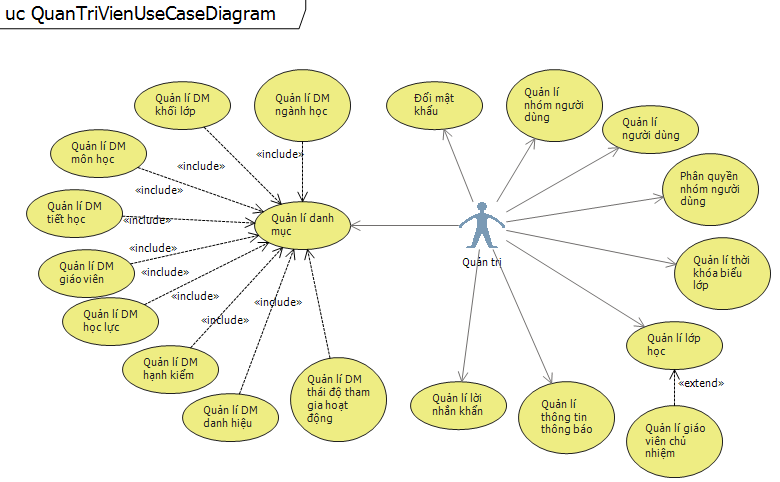
### 3.4.1. Use Case Diagram



**Hình – Diagram của actor Phụ huynh**



**Hình – Diagram của actor Giáo viên**



**Hình – Diagram của actor Quản trị**

### 3.4.2. Design

#### 3.4.2.1. Class Design Diagram

#### 3.4.2.2. Screen Design

#### 3.4.2.3. Database Design

## 3.5. Kết quả thực hiện

# 4. Kế hoạch

Kế hoạch tổng quan:

|  |  |  |  |
| --- | --- | --- | --- |
| **Tuần** | **Thời gian** | **Nội dung** | **Chi tiết** |
| 1 | 19.9 - 24.9 | Nghiên cứu  công nghệ điện toán đám mây | Tổng quan công nghệ ĐTĐM Windows Azure |
| 2 | 26.9 - 1.10 | Dịch vụ lưu trữ ĐTĐM Windows Azure Service |
| 3 | 3.10 - 8.10 | Cơ sở dữ liệu quan hệ ĐTĐM SQL Azure |
| 4 | 10.10 - 15.10 | Tài liệu hóa nghiên cứu |
| 5 | 17.10 - 22.10 | Khảo sát thực tế, thu thập yêu cầu,  nghiên cứu khả năng | Xây dựng bài toán Sổ liên lạc trực tuyến |
| 6 | 24.10 - 29.10 | Thu thập yêu cầu, nghiệp vụ |
| 7 | 31.10 - 5.11 | Cài đặt | Use Case, Class Diagram |
| 8 | 7.11 - 12.11 | Screen Diagram, Database Diagram |
| 9 | 14.11 - 19.11 | Coding |
| 10 | 21.11 - 26.11 |
| 11 | 28.10 - 3.12 |
| 12 | 5.12 - 10.12 | Kiểm thử | Test, Fix bug |
| 13 | 12.12 - 17.12 |
| 14 | 19.12 - 24.12 |

# 5. Kết luận và hướng phát triển