# Giới thiệu

* VNTT
* Dịch vụ Cloudbackup:

# Nhu cầu

## Hiện trạng ứng dụng portal

## Các chức năng

Nội dung công việc

## Các vấn đề

* Số lượng dữ liệu cho dashboard: Với mỗi tài khoản manager sẽ có nhiều tài khoản admin, mỗi tài khoản admin lại có nhiều backupset. Do đó số lượng backupset trong mỗi tài khoản admin là khá lớn. Trong một tháng( hoặc một ngày) được hiển thị trên dashboard, số lượng các dòng(record) lưu thông tin của backup job là rất lớn.

Ví dụ: với 1 tài khoản manager có 5 tài khoản vguard pro và 5 tài khoản vguard lite, mỗi tài khoản có 5 backupset (riêng vguard lite chỉ có 1 backupset). Số lượng backupset là: 30 backupset. Mỗi backupset này có thể thực hiện nhiều lần backup dữ liệu. Cụ thể theo thiết kế, có thể lập lịch cho backupset thực hiện backupset mỗi 2 phút. Tuy nhiên, với điều kiện mỗi backupset thực hiện backup 1 lần/4h thì số backupjob là 180 backupjob.

Theo trung bình, nếu có 20 user thì số backupjob mỗi ngày là 180 x 20 = 36000 backupjob, một con số rất lớn.

* Tốc độ obs: Thời gian trả lời của obs khi truy xuất các record là khá lâu và API của Obs chỉ hỗ trợ truy xuất backup Job trên từng tài khoản admin vào một ngày xác định. Ngoài ra các thông tin cung cấp bởi chức năng truy xuất trên không thể cung cấp đầy đủ thông tin cho các backup job mà đòi hỏi với mỗi backupjob cần có một câu truy vấn obs khác để lấy thông tin. Tất cả dẫn đến thời gian truy vấn trực tiếp trên Obs để lấy dữ liệu về backupJob là rất dài.

Cụ thể: nếu muốn truy cập danh sách backupjob của một tài khoản admin trong một tháng thì phải thực hiện 30 truy vấn đến obs (giả định tháng có 30 ngày). Mỗi backupjob trong một ngày cần thực hiện thêm 2 lần truy vấn obs để có thông tin đầy đủ (truy vấn theo ngày chỉ trả về status) như danh sách files, tổng dung lượng thay đổi, thời gian bắt đầu, kết thúc… do đó, nếu có để thực hiện truy vấn được 36000 backupjob thì sẽ tốn rất nhiều thời gian, không thể đáp ứng nếu truy xuất trực tiếp trên obs.

* Vấn đề API: API của ashay cung cấp backupset được tạo theo thời gian (ID tính theo millisecond từ 1/1/1970) nên vẫn có thể trùng ID backupset. Một khó khăn khác là cập nhật dữ liệu từ obs để lưu trữ trên database, trong một số lần chạy demo thì xuất hiện có những backupjob cùng ID của cùng backupset nhưng khác status. Hai backupjob trên có một status là missed. Do đó, việc cập nhật dữ liệu và xác nhận, kiểm tra để thêm record vào database gặp khó khăn.
* Vấn đề về cập nhật: Do thời gian chạy backupjob trên các user là khác nhau, có thể có những backupjob chạy rất lâu, đăc biệt là chạy lần đầu khi mới tạo backupset. Do không có thông tin về lần cuối cùng chạy backupjob nên phải update từ lúc có backup chưa hoàn tất. Tuy nhiên số lượng backupjob lại rất lớn nên nếu chạy lại nhiều ngày sẽ rất chậm nên cần xác định thời gian giới hạn để cập nhật.
* Tần suất xem: Các thông tin về tài khoản (manager, admin) , backupset được xem với tần số thấp và có thể được thay đổi trong một phiên nên không được lưu trữ trên web mà được truy vấn với mỗi khi có yêu cầu xem từ phí người dung.
* Vấn đề bất đồng bộ
* Kiểm tra chất lượng các hàm truy vấn Obs: Dữ liệu được truy vấn từ Obs thông qua các hàm truy vấn API của Assay, do đó có 40 hàm truy vấn dữ liệu khác nhau. Bên cạnh đó dữ liệu API trả về dưới dạng XML phải parse dữ liệu thành Object để lưu lại xử lý cho các tác vụ khác, vì vậy số lượng các lớp chứa dữ liệu cũng rất lớn. Vì đây là dữ liệu được sử dụng cho toàn chương trình nên nó yêu cầu phải chính xác tuyệt đối, mỗi hàm xử lý đều được thiết kế Unit Tests trước khi hiện thực, Unit tests được thực hiện chia làm 2 lớp khác nhau: kiểm tra dữ liệu parse từ XML về các lớp và dữ liệu được kiểm tra lại ở các services. Các test cases này được chạy thường xuyên trong và sau quá trình hiện thực để dữ liệu luôn đảm bảo.

## Công nghệ:

* Unit Test

1. Định nghĩa.
   * Unit Test là quá trình kiểm thử các thành phần của chương trình như các phương thức, các lớp đối tượng hoặc các đơn vị khác trong code. Unit Test sẽ giúp developers có khả năng để xác định chức năng của họ hoạt động đúng như mong đợi.
2. Lợi ích của Unit Test.

* Unit Tests giúp xác định giải thuật, logic không đúng hoặc các lỗi trong quá trình lập trình từ sớm để sửa chữa và nâng cao chất lượng code.
* Unit Tests giúp có thể kiểm tra lại code một cách nhanh chóng và chính xác khi thay đổi một phần nhỏ trong chương trình.
* Unit Tests chứng minh rằng code của bạn đang hoạt động.
* Unit Tests giúp cải thiện thiết kế code mà không phá vỡ cấu trúc code.
* Test-first buộc developers phải có kế hoạch trước khi lập trình.
* Chi phí và thời gian hiện thực phần mềm giảm, bên cạnh đó chất lượng code được nâng cao và giảm rủi ro của từng chức năng một trong phần mềm.

1. Giới thiệu một số Framework Unit tests trên các ngôn ngữ khác nhau.
2. C#
   * + Visual studio: Visual Studio Unit Testing Framework lần đầu tiên được bao gồm trong Visual Studio Team System 2005, nơi nó được tích hợp với IDE, nhưng không có sẵn trong hầu hết được sử dụng Standard Edition. Từ Visual Studio 2008 nó có sẵn cũng Professional Edition. Bắt đầu với Visual Studio Express 2013, nó được bao gồm các phiên bản với Visual Studio Express.
3. Java
   * + JUnit là một framework đơn giản để viết bài kiểm tra lặp lại. Nó là một hiện thực của kiến ​​trúc xUnit cho các khuôn khổ kiểm tra đơn vị.
4. C/C++
   * + Parasoft C/C++ test: Mô phỏng con đường thực hiện ứng dụng khả thi trong nhiều chức năng và dữ liệu là rất quan trọng cho các ứng dụng nhúng trong đó phân tích chi tiết cho thời gian chạy là không hiệu quả hoặc không thể. C / C ++ kiểm tra cũng cung cấp một đường dẫn đầy đủ cho mỗi khiếm khuyết tiềm năng trong IDE và qua các liên kết nó với code, cho phép người dùng nhanh chóng chuyển đến bất kỳ điểm nào trong đường dẫn phân tích nhấn mạnh. Các lỗi được phát hiện bao gồm sử dụng chưa được khởi tạo hoặc không hợp lệ bộ nhớ, null dereferencing con trỏ, mảng và tràn bộ đệm, chia cho không, bộ nhớ và tài nguyên bị rò rỉ, và các lỗi khác nhau làm code không hoạt động.
5. JavaScript
   * + QUnit: là một framework Javascript unit testing mạnh mẽ và dễ dàng sử dụng Nó được sử dụng bởi các dự án jQuery, jQuery UI và jQuery Mobile và có khả năng kiểm tra bất kỳ mã JavaScript.
6. Unit tests trong portal.
   * Để đạt độ chính xác cao, tiết kiệm chi phí, thời gian và nâng cao tính chuyên nghiệp trước mỗi đoạn code được viết đều được thiết kế và hiện thực các test cases tự động trước. Các dữ liệu trong database được sử dụng kĩ thuật mock để giả lập cho các unit tests để đảm bảo hạn chế lỗi tối đa khi chạy trong môi trường thực tế. Bên cạnh đó, số lượng test cases nhiều đảm bảo các chức năng chạy đúng như mong muốn do đó số lượng bug được tìm ra và sửa chữa sớm. Trong quá trình sửa chữa, các test cases luôn được chạy lại để đảm bảo cấu trúc của chương trình không bị phá vỡ.
7. Công việc thực hiện.

Bước 1: Tạo một kế hoạch thử nghiệm   
Bước 2: Tạo các test cases và test data   
Bước 3: Nếu có thể tạo ra các scripts để chạy các test cases.   
Bước 4: Chạy test cases cho mỗi khi hoàn thành code.   
Bước 5: Sửa chữa các lỗi nếu có và chạy lại test cases.   
Bước 6: Lặp lại chu kỳ test cho đến khi "đơn vị" không còn lỗi.

* Bất đồng bộ trên giao diện
* SQL:
* Lưu trữ: Đây là chìa khóa để giải quyết bài toàn số lượng dữ liệu lớn và tốc độ truy vấn trên obs khi thực hiện dashboard. Để có thể truy vấn dữ liệu trên dashboard nhanh chóng, những thông tin về backupjob được lưu trữ trực tiếp trên database của portal. Bảng backupjob trong database bao gồm thông tin của truy vấn status và truy vấn report (để lấy thông tin chi tiết) nên dễ dàng tổng truy xuất. Ngoài ra, database có thời gian truy vấn nhanh hơn obs rất nhiều nên giảm được thời gian.
* Truy vấn:

## Làm việc nhóm

* Team Foudation

# Kết quả

## Đạt được

## Hạn chế

Tổng quát hóa phân tích dữ liệu

1. Định nghĩa

Các dữ liệu trả về ban đầu là khối lượng lớn dữ liêu thô, không có tổ chức và được lấy từ nhiều nguồn khác nhau, do đó cần một quá trình phân tích, biến đổi tổng quát các dữ liệu này thành các đối tượng để sử dụng một cách thuận tiện nhất trong các tác vụ khác liên quan đến dữ liệu.

1. Vấn đề

* Toàn bộ dữ liệu của chương trình được truy xuất thông qua các Obs server, dữ liệu gồm thông tin về users, backup set, backup job và các thông tin về hệ thống. Tuy nhiên các thông tin này được chia làm các API khác nhau và các services được viết trên các API tạo nên 32 services khác nhau. Bên cạnh số lượng services lớn mỗi service lại có các thông số đầu vào (paramaters) riêng và trả về các dữ liệu riêng. Dẫn đến việc tạo riêng các paramaters và các objects khác nhau để cho từng service sẽ tốn một chi và thời gian lớn. Ngoài ra, điều này sẽ làm quá trình kiểm tra tính đúng đắn dữ liệu khó khăn và khó kiểm soát trong các tác vụ liên quan đến dữ liệu.
* Dữ liệu các servers Obs cung cấp ở dạng dữ liệu thô XML với số lượng thuộc tính lớn, do đó việc này sẽ gây khó khăn trong quá trình kiểm tra lại dữ liệu khi truy xuất. Ngoài ra cấu trúc XML khá phức tạp gồm nhiều tập hợp và đối tượng ở các node phía trong và mỗi API lại trả cấu trúc khác do đó không thể áp dụng được các công nghệ serialization và deserialization XML của C#.
* Ngoài những khó khăn trên, trong dữ liệu trên Obs có một số dữ liệu cần được truy xuất thường xuyên, nếu không được xứ lý thích hợp chi phí và thời gian bỏ trong quá trình truy xuất lên Obs sẽ gây ra ảnh hưởng lớn tốc độ của chương trình.

1. Phân tích

* Tuy dữ khối lượng các thông số đầu vào và dữ liệu trả về của service rất lớn nhưng nó có nhiều thuộc tính chung và có một số service lấy dữ liệu cho chung một đối tượng.
* Dữ liệu XML trả về phức tạp nhưng cấu trúc nó chỉ lặp lại ở 1 số dạng cấu trúc nhất định, có thể xử lí linh hoạt để đọc dữ liệu.
* Các dữ liệu thường xuyên được sử dụng nhưng không phải lúc nào thông tin đó cũng thay đổi nên ta có thể sử dụng một số kĩ thuật để kiểm tra và truy xuất khi cần thiết.

1. Giải pháp

* Phân tích tổng quát thành các đối tượng chính để dùng chung các services có các dữ liệu trả về chung cho một đối tượng,ví dụ backup set, user profile,… . Tương tự với dữ liệu đầu ta sẽ tạo một lớp gồm các đối tượng chung và các lớp khác sẽ thừa kế lại những lớp chung này. Tuy nhiên khi tạo các lớp này số lượng các thuộc tính vẫn còn lớn để đảm bảo chất lượng không sai xót khi tạo các lớp này, có một giải pháp khá hữu ích là viết một chương trình đơn giản để tự động lọc ra các tên thuộc tính của các API do Obs cung cấp và chuyển thành định dạng thuộc tính trong C#. Việc này giảm thời gian đáng kể và chất lượng các tên thuộc tính đảm bảo.
* Sau khi phân tích các dữ liệu XML gồm ba cấu trúc chính Object, Collection,Primitive và kết hợp của chúng, do đó ta tách ra làm 3 hàm con để xử lý riêng cho các dạng dữ liệu. Các generic method được viết để tăng tính linh hoạt phù hợp với tất cả các dữ liệu trả về. Bên cạnh đó một số kĩ thuật về tạo đối tượng bằng Activator Class để giải quyết vấn đề của các Objects và Collections.
* Các dữ liệu cần thiết ví dụ như thông tin Users cần được update kiểm tra liên tục, do đó chúng ta sẽ tạo ra cache để lưu lại danh sách users và chúng ta sẽ phân tích tính toán thời gian hợp lý để update lại danh sách. Do đó danh sách này chỉ được cập nhật khi User được kiểm tra chưa nằm trong danh sách hoặc sau khoảng thời gian nhất định. Ngoài ra, chúng ta có một phương thức force update để dành cho một số trường hợp như tạo tài khoản.