

TRƯỜNG ĐẠI HỌC SƯ PHẠM TP.HCM
Khoa Công Nghệ Thông Tin

BÁO CÁO CƠ SỞ DỮ LIỆU NÂNG CAO

CHỦ ĐỀ: TÌM HIỂU VỀ REDIS

Giảng Viên: Lương Trần Hy Hiến

Nhóm thực hiện: 21

- 1. Trần Thị Hồng Hà**
- 2. Nguyễn Thị Thanh Ngân**
- 3. Bang Minh Kiệt**
- 4. Phan Thanh Hữu**
- 5. Hoàng Phi Long**

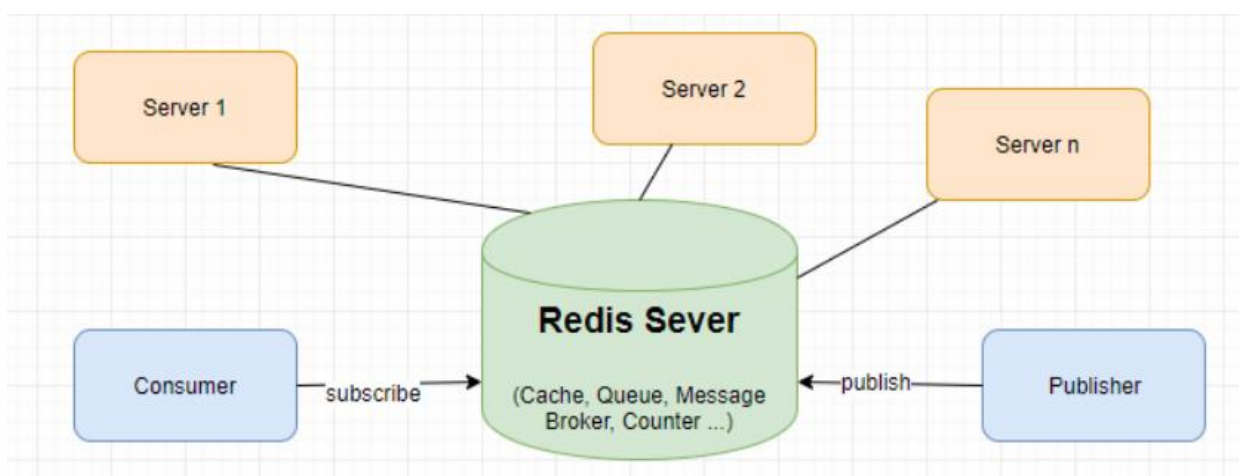
Thành phố Hồ Chí Minh – Năm 2019

CHƯƠNG 1 - TỔNG QUAN

1.1. Khái niệm và Đặc điểm

1.1.1. Khái niệm

Redis (Remote Dictionary Server) là cơ sở dữ liệu NoSQL, một mã nguồn mở được dùng để lưu trữ dữ liệu có cấu trúc, có thể sử dụng như một database, bộ nhớ cache một message broker hay danh sách các bộ chờ xử lý. Nó là hệ thống lưu trữ dữ liệu với dạng KEY-VALUE rất mạnh mẽ và phổ biến hiện nay.



Nguồn: <https://topdev.vn/blog/redis-la-gi/>

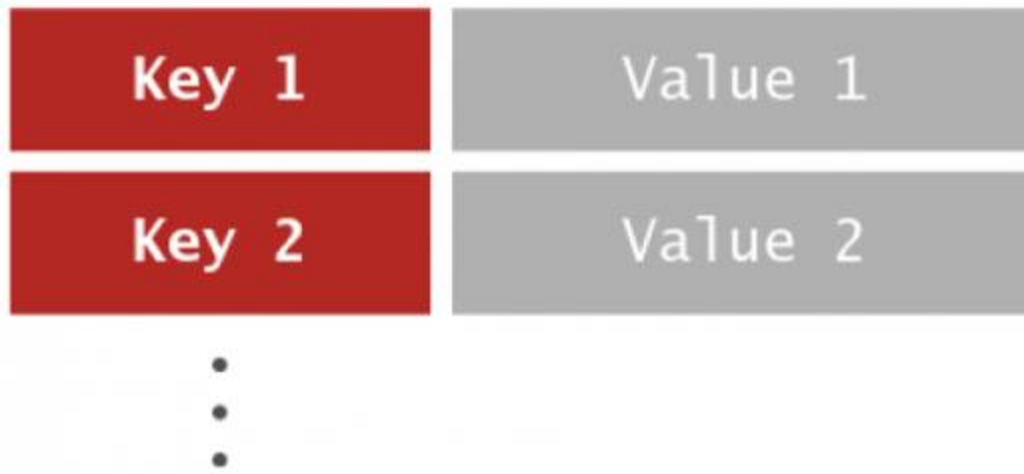
1.1.2. Đặc điểm

- Redis còn là một loại in-memory storage, tức là toàn bộ database của nó được lưu tại RAM thay vì ổ cứng.
- Số lượng data types được định nghĩa trong Redis nhiều hơn hẳn so với các loại key-value storage khác.
- Redis có thể sao chép data sang nhiều slaves khác nhau.

1.2. Các kiểu dữ liệu trong Redis

Khác với RDMS như MySQL, hay PostgreSQL, Redis không có table (bảng). Redis lưu trữ data dưới dạng key-value. Thực tế thì memcache cũng làm vậy, nhưng kiểu dữ liệu của memcache bị hạn chế, không đa dạng được như Redis, do đó không hỗ trợ được nhiều thao tác từ phía người dùng. Dưới đây là sơ lược về các kiểu dữ liệu Redis dùng để lưu value.

- **STRING**: string, integer hoặc float. Redis có thể làm việc với cả string, từng phần của string, cũng như tăng/giảm giá trị của integer, float.



Hình 1. Minh họa String

- **LIST**: List là một danh sách của strings, sắp xếp theo thứ tự insert. Redis có thể thêm một phần tử vào đầu hoặc cuối list. List phù hợp cho các bài toán cần thao tác với các phần tử gần đầu và cuối vì việc truy xuất này là cực nhanh, cho dù insert cả triệu phần tử. Tuy nhiên nhược điểm là việc truy cập vào các phần tử ở giữa list rất chậm.



Hình 2. Minh họa cho List

- **SET**: tập hợp các string (không được sắp xếp). Redis hỗ trợ các thao tác thêm, đọc, xóa từng phần tử, kiểm tra sự xuất hiện của phần tử trong tập hợp. Ngoài ra Redis còn hỗ trợ các phép toán tập hợp, gồm intersect/union/difference.

Key A	{Member 1, Member 4, Member 2, Member 3, Member 5} ...
Key B	{Apples; Pears; Shoes; Candy; Bage}s} ...

Hình 3. Minh họa cho Set

- **HASH**: lưu trữ hash table của các cặp key-value, trong đó key được sắp xếp ngẫu nhiên, không theo thứ tự nào cả. Redis hỗ trợ các thao tác thêm, đọc, xóa từng phần tử, cũng như đọc tất cả giá trị.

Key A		Key B		Key C	
Field 1	Value A	Field 1	Value X	Field 1	Value P
Field 2	Value B	Field 2	Value Y	Field 2	Value Q
Field 3	Value C		⋮	Field 3	Value R
Field 4	Value D				⋮
	⋮				

Hình 4. Minh họa cho Hash

- **SORTED SET (ZSET)**: là 1 danh sách, trong đó mỗi phần tử là map của 1 string (member) và 1 floating-point number (score), danh sách được sắp xếp theo score này. Các phần tử của zset được sắp xếp theo thứ tự từ score nhỏ tới lớn.

Key A	
Member R	Score=300
Member X	Score=500
Member P	Score=1000
⋮	

Hình 5: Minh họa cho Sorted Set

Ngoài ra, Redis còn hỗ trợ các data types khác như: Bit arrays, HyperLogLogs, Streams.

1.3. Cách thức hoạt động của Redis và Redis Persistence

1.3.1. Cách thức hoạt động

Toàn bộ dữ liệu Redis nằm trong bộ nhớ, trái với cơ sở dữ liệu thường lưu dữ liệu trên ổ đĩa hoặc ổ SSD. Bằng cách loại bỏ sự cần thiết phải truy cập ổ đĩa, kho dữ liệu trong bộ nhớ như Redis tránh được sự chậm trễ do thời gian tìm kiếm và có thể truy cập dữ liệu trong vài micro giây. Redis có cấu trúc dữ liệu linh hoạt, độ khả dụng cao, dữ liệu không gian địa lý, ngôn ngữ kịch bản Lua, giao dịch, lưu trữ lâu dài trên ổ đĩa và hỗ trợ cụm, giúp xây dựng các ứng dụng quy mô Internet theo thời gian thực dễ dàng hơn.

1.3.2. Redis Persistence

Bên cạnh việc lưu key-value trên bộ nhớ RAM, Redis có 2 background threads chuyên làm nhiệm vụ định kỳ ghi dữ liệu lên đĩa cứng.

Có 2 loại file được ghi xuống đĩa cứng: **RDB (Redis DataBase file)** và **AOF (Append Only File)**

- **RDB (Redis DataBase file)** Cách thức làm việc RDB thực hiện tạo và sao lưu snapshot của DB vào ổ cứng sau mỗi khoảng thời gian nhất định.

Ưu điểm

RDB cho phép người dùng lưu các version khác nhau của DB, rất thuận tiện khi có sự cố xảy ra. Bằng việc lưu trữ data vào 1 file cố định, người dùng có thể dễ dàng chuyển data đến các data centers, máy chủ khác nhau. RDB giúp tối ưu hóa hiệu năng của Redis. Tiến trình Redis chính sẽ chỉ làm các công việc trên RAM, bao gồm các thao tác cơ bản được yêu cầu từ phía client như thêm/đọc/xóa, trong khi đó 1 tiến trình con sẽ đảm nhiệm các thao tác disk I/O. Cách tổ chức này giúp tối đa hiệu năng của Redis. Khi restart server, dùng RDB làm việc với lượng data lớn sẽ có tốc độ cao hơn là dùng AOF.

Nhược điểm

RDB không phải là lựa chọn tốt nếu bạn muốn giảm thiểu tối đa nguy cơ mất mát dữ liệu. Thông thường người dùng sẽ set up để tạo RDB snapshot 5 phút 1 lần (hoặc nhiều hơn). Do vậy, trong trường hợp có sự cố, Redis không thể hoạt động, dữ liệu trong những phút cuối sẽ bị mất. RDB cần dùng fork() để tạo tiến trình con phục vụ cho thao tác disk I/O. Trong trường hợp dữ liệu quá lớn, quá trình fork() có thể tốn thời gian và server sẽ không thể đáp ứng được request từ client trong vài milisecond hoặc thậm chí là 1 second tùy thuộc vào lượng data và hiệu năng CPU.

- **AOF (Append Only File)** Cách thức làm việc AOF lưu lại tất cả các thao tác write mà server nhận được, các thao tác này sẽ được chạy lại khi restart server hoặc tái thiết lập dataset ban đầu.

Ưu điểm

Sử dụng AOF sẽ giúp đảm bảo dataset được bền vững hơn so với dùng RDB. Người dùng có thể config để Redis ghi log theo từng câu query hoặc mỗi giây 1 lần. Redis ghi log AOF theo kiểu thêm vào cuối file sẵn có, do đó tiến trình seek trên file có sẵn là không cần thiết. Ngoài ra, kể cả khi chỉ 1 nửa câu lệnh được ghi trong file log (có thể do ổ đĩa bị full), Redis vẫn có cơ chế quản lý và sửa chữa lỗi đó (redis-check-aof). Redis cung cấp tiến trình chạy nền, cho phép ghi lại file AOF khi dung lượng file quá lớn. Trong khi server vẫn thực hiện thao tác trên file cũ, 1 file hoàn toàn mới được tạo ra với số lượng tối thiểu operation phục vụ cho việc tạo dataset hiện tại. Và 1 khi file mới được ghi xong, Redis sẽ chuyển sang thực hiện thao tác ghi log trên file mới.

Nhược điểm

File AOF thường lớn hơn file RDB với cùng 1 dataset. AOF có thể chậm hơn RDB tùy theo cách thức thiết lập khoảng thời gian cho việc sao lưu vào ổ cứng. Tuy nhiên, nếu thiết lập log 1 giây 1 lần có thể đạt hiệu năng tương đương với RDB. Developer của Redis đã từng gặp phải bug với AOF (mặc dù là rất hiếm), đó là lỗi AOF không thể tái tạo lại chính xác dataset khi restart Redis. Lỗi này chưa gặp phải khi làm việc với RDB bao giờ.

1.4. Ưu điểm và Nhược điểm của Redis

- **Ưu điểm**
 - **Kho dữ liệu trong bộ nhớ:** Toàn bộ dữ liệu Redis nằm trong bộ nhớ chính của máy chủ, trái với cơ sở dữ liệu, chẳng hạn như PostgreSQL, Cassandra, MongoDB, v.v. thường lưu phần lớn dữ liệu trên ổ đĩa hoặc ổ SSD. So với cơ sở dữ liệu trên ổ đĩa truyền thống trong đó phần lớn các tác vụ đều yêu cầu truy cập qua lại tới ổ đĩa, kho dữ liệu trong bộ nhớ chẳng hạn như Redis không phải chịu hình phạt này. Do đó kho dữ liệu kiểu này có thể hỗ trợ thêm được khá nhiều tác vụ và có thời gian phản hồi nhanh hơn. Kết quả là – hiệu suất nhanh thấy rõ với các tác vụ đọc hoặc ghi thông thường mất chưa đầy một mili giây và hỗ trợ hàng triệu tác vụ mỗi giây.
 - **Tốc độ nhanh rõ rệt:** do truy xuất dữ liệu trên RAM, tốc độ query dữ liệu của Redis là cực kỳ nhanh, có thể lên đến 110000 lệnh SETs mỗi giây, 81000 GETs mỗi giây.
 - **Redis hỗ trợ “Multiple Database”** với nhiều commands để tự động remove key từ một database tới database khác
 - **Hỗ trợ nhiều loại data types khác nhau:** Redis hỗ trợ hầu hết các loại data types được các developers hay dùng hiện nay như list, set, sorted set, và hashes. Điều này giúp ta dễ dàng hơn để giải quyết nhiều bài toán khác nhau khi mà một loại data type cụ thể nào đó mạnh/dễ dùng hơn các loại khác.
 - **Đơn giản và dễ sử dụng:** Redis đơn giản hóa mã bằng cách cho phép bạn viết ít dòng lệnh hơn để lưu trữ, truy cập và sử dụng dữ liệu trên ứng dụng của bạn. Ví dụ: nếu ứng dụng của bạn có dữ liệu được lưu trên một bảng băm và bạn muốn lưu dữ liệu đó trên kho dữ liệu – bạn chỉ cần sử dụng cấu trúc dữ liệu mã hash của Redis để lưu dữ liệu đó. Tác vụ tương tự trên kho dữ liệu không có cấu trúc dữ liệu mã hash sẽ cần nhiều dòng mã để chuyển đổi từ định dạng này sang định dạng khác. Redis được trang bị cấu trúc dữ liệu riêng và nhiều tùy chọn để điều khiển và tương tác với dữ liệu của bạn. Trên một trăm máy khách mã nguồn mở được

cung cấp cho nhà phát triển Redis. Các ngôn ngữ được hỗ trợ gồm có Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby, R, Go và nhiều ngôn ngữ khác.

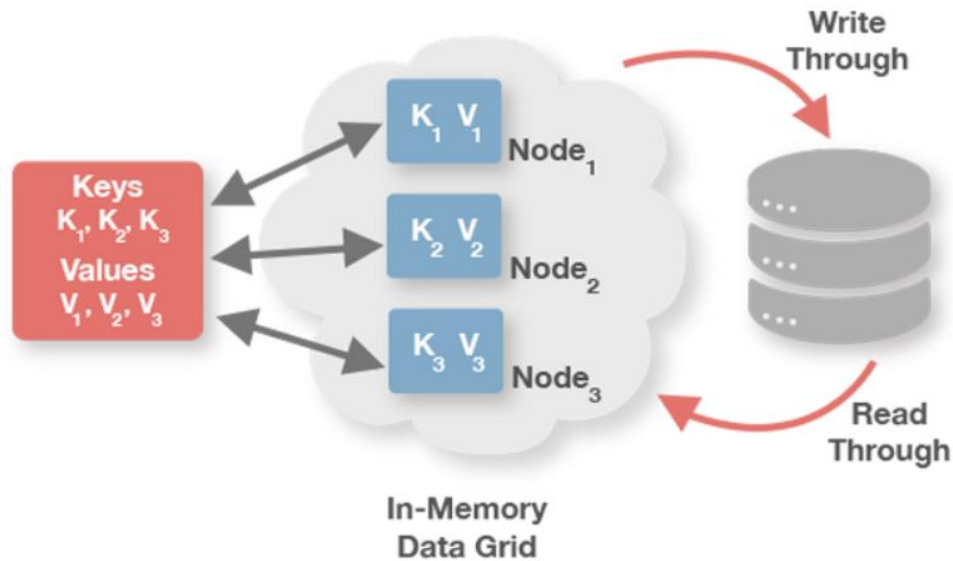
- ***Dù là NOSQL database, nhưng các operation ở Redis vẫn có tính chất atomic***, tức là nếu có từ 2 clients trở lên cùng vào database, Redis server sẽ luôn cập nhật giá trị data mỗi khi có thay đổi.
- ***Redis có thể được dùng trong nhiều use-cases khác nhau*** chẳng hạn như caching, messaging-queues (Redis hỗ trợ Publish/Subscribe một cách native), các data có vòng đời ngắn trong application như sessions, hit counts...
- ***Khả năng mở rộng***: Redis là dự án mã nguồn mở được một cộng đồng đông đảo ủng hộ. Không có giới hạn về nhà cung cấp hoặc công nghệ vì Redis được có tính tiêu chuẩn mở, hỗ trợ các định dạng dữ liệu mở và có tập hợp máy khách phong phú.
- **Nhược điểm**
 - Vì Redis sử dụng RAM làm bộ nhớ cho mình nên khi lượng file cache lớn thì sẽ dẫn đến trường hợp thiếu RAM cho Server. Đây là nhược điểm lớn nhất của Redis
 - Tính linh hoạt của CSDL dạng key – value bị đánh đổi bởi tính chính xác. Hầu như rất khó để truy xuất giá trị chính xác từ CSDL dạng này vì dữ liệu được lưu trữ theo **BLOB (Binary large object)**, nên kết quả trả về hầu như đều theo **BLOB**. Điều này gây ra khó khăn khi báo cáo số liệu hoặc cần chỉnh sửa một phần của các giá trị. Cuối cùng, không phải objects nào cũng có thể được cấu hình thành cặp chìa khoá – giá trị được.

1.5. Các ứng dụng và trường hợp sử dụng của Redis

- **Ứng dụng**

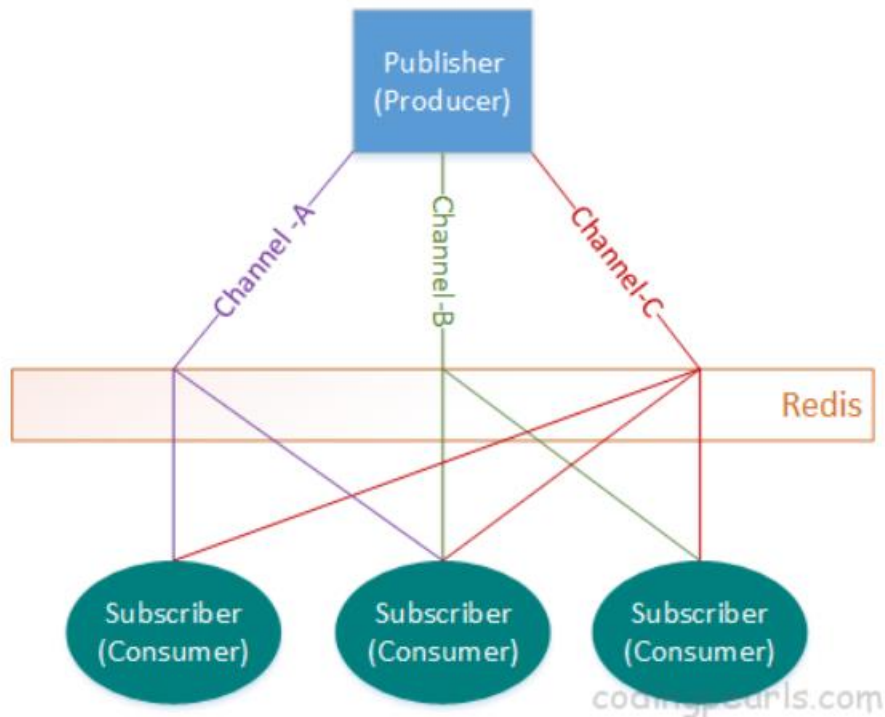
Redis ngoài tính năng lưu trữ KEY-VALUE trên RAM thì Redis còn hỗ trợ tính năng sắp xếp, query, backup dữ liệu trên đĩa cứng cho phép bạn có thể phục hồi dữ liệu khi hệ thống gặp sự cố...và có thể nhân bản (Chạy nhiều Server Redis cùng lúc).

- ***Caching***: Sử dụng làm bộ nhớ đệm. Chính tốc độ đọc ghi nhanh mà Redis có thể làm bộ nhớ đệm, nơi chia sẻ dữ liệu giữa các ứng dụng hoặc làm database tạm thời. Ngoài ra Redis có thể sử dụng để làm Full Page Cache cho website. Cũng vì tính nhất quán của Redis, cho dù restart Redis thì người dùng cũng không có cảm nhận chậm khi tải trang.



Nguồn. https://github.com/phamtai97/Report_Cache

- **Counter:** Sử dụng làm bộ đếm. Với thuộc tính tăng giảm thông số rất nhanh trong khi dữ liệu được lưu trên RAM, sets và sorted sets được sử dụng thực hiện đếm lượt view của một website, các bảng xếp hạng trong game chẳng hạn. Redis hỗ trợ thread safe do đó nó có thể đồng bộ dữ liệu giữa các request.
- **Publish/Suscribe (Pub/Sub):** Tạo kênh chia sẻ dữ liệu. Redis hỗ trợ tạo các channel để trao đổi dữ liệu giữa publisher và subscriber giống như channel trong Socket Cluster hay topic trong Apache Kafka. Ví dụ: Pub/Sub được sử dụng theo dõi các kết nối trong mạng xã hội hoặc các hệ thống chat.



- **Queues:** Tạo hàng đợi để xử lý lần lượt các request. Redis cho phép lưu trữ theo list và cung cấp rất nhiều thao tác với các phần tử trong list, vì vậy nó còn được sử dụng như một message queue.
- **Các trường hợp sử dụng phổ biến Redis**
- **Lưu trữ bộ nhớ đệm**

Redis là lựa chọn tuyệt vời để triển khai một bộ nhớ đệm trong bộ nhớ có độ khả dụng cao để giảm độ trễ truy cập dữ liệu, tăng năng suất và giảm tải cho cơ sở dữ liệu quan hệ hoặc NoSQL và ứng dụng của bạn. Redis có thể phục vụ những mục dữ liệu thường xuyên được yêu cầu với thời gian phản hồi chưa đến một mili giây và cho phép bạn dễ dàng thay đổi quy mô nhằm đáp ứng mức tải cao hơn mà không cần gia tăng phần backend có chi phí tốn kém hơn. Một số ví dụ phổ biến về nhớ đệm khi sử dụng Redis bao gồm nhớ đệm kết quả truy vấn cơ sở dữ liệu, nhớ đệm phiên lâu bền, nhớ đệm trang web và nhớ đệm các đối tượng thường xuyên được sử dụng như ảnh, tập tin và siêu dữ liệu.

- **Trò chuyện, nhắn tin và danh sách tác vụ chờ xử lý**

Redis hỗ trợ Pub/Sub (cấu trúc gửi nhận tin nhắn trong đó người gửi và người nhận không biết nhau) với tính năng khớp cấu trúc và nhiều cấu trúc dữ liệu như danh sách, tập được sắp xếp và mã hash. Việc này cho phép Redis hỗ trợ các phòng trò chuyện

hiệu suất cao, luồng bình luận theo thời gian thực, nguồn cấp mạng xã hội và giao tiếp giữa các máy chủ. Cấu trúc dữ liệu Danh sách của Redis giúp dễ dàng triển khai một danh sách tác vụ chờ xử lý có tải trọng nhẹ. Danh sách cung cấp các hoạt động nguyên tử cũng như tính năng chặn, giúp cho chúng phù hợp với nhiều ứng dụng yêu cầu phải có trình chuyển tiếp tin nhắn tin cậy hoặc danh sách liên kết vòng.

- ***Bảng xếp hạng game***

Redis là giải pháp hay được các nhà phát triển game dùng để xây dựng bảng xếp hạng theo thời gian thực. Chỉ cần sử dụng cấu trúc dữ liệu Tập được sắp xếp của Redis, cấu trúc dữ liệu này đảm bảo tính duy nhất của các thành phần trong khi vẫn duy trì danh sách được sắp xếp theo điểm số của người dùng. Tạo danh sách xếp hạng theo thời gian thực dễ thực hiện như khi cập nhật điểm số của người dùng mỗi khi có thay đổi. Bạn cũng có thể sử dụng Tập được sắp xếp để xử lý dữ liệu chuỗi thời gian bằng cách dùng dấu thời gian làm điểm số.

- ***Kho lưu trữ phiên***

Redis là kho dữ liệu trong bộ nhớ có độ khả dụng và độ bền cao, thường được các nhà phát triển ứng dụng sử dụng để lưu trữ và quản lý dữ liệu phiên cho các ứng dụng quy mô internet. Redis có độ trễ chưa đến một mili giây, có quy mô và độ đàn hồi cần thiết để quản lý dữ liệu phiên chẳng hạn như hồ sơ người dùng, thông tin xác thực đăng nhập, trạng thái phiên và tùy chỉnh theo ý muốn người dùng.

- ***Phát nội dung giàu dữ liệu***

Redis cung cấp kho dữ liệu trong bộ nhớ, có tốc độ truy cập nhanh để đáp ứng các trường hợp sử dụng phát trực tiếp. Có thể sử dụng Redis để lưu trữ siêu dữ liệu về hồ sơ người dùng và xem lịch sử, thông tin/mã thông báo xác thực cho hàng triệu người dùng và hiển thị tập tin để cho phép các Mạng truyền tải nội dung (CDN) phát video cho hàng triệu người dùng di động và máy tính để bàn cùng một lúc.

- ***Dữ liệu không gian địa lý***

Redis cung cấp cấu trúc dữ liệu trong bộ nhớ, được tích hợp sẵn cho mục đích cụ thể và các toán tử để quản lý dữ liệu không gian địa lý theo thời gian thực ở quy mô và tốc độ mong muốn. Các lệnh như GEOADD, GEODIST, GEORADIUS và GEORADIUSBYMEMBER để lưu trữ, xử lý và phân tích dữ liệu không gian địa lý theo thời gian thực giúp cho dữ liệu không gian địa lý trở nên dễ dàng và nhanh chóng khi sử dụng Redis. Bạn có thể sử dụng Redis để thêm các tính năng dựa trên địa điểm như thời gian lái xe, quãng đường lái xe và các điểm quan tâm cho ứng dụng của bạn.

- **Machine Learning**

Các ứng dụng kiểu mới, chịu sự chi phối của dữ liệu yêu cầu machine learning phải có khả năng nhanh chóng xử lý được dữ liệu theo khối lượng lớn, đa dạng, tốc độ cao và tự động hóa quá trình ra quyết định. Đối với các trường hợp sử dụng như phát hiện lỗi trong các dịch vụ game và tài chính, đấu thầu theo thời gian thực trong công nghệ quảng cáo và mai mối trong hẹn hò và đi chung xe, khả năng xử lý dữ liệu trực tiếp và ra quyết định trong vòng vài chục mili giây có ý nghĩa hết sức quan trọng. Redis cung cấp cho bạn kho dữ liệu trong bộ nhớ, có tốc độ truy cập nhanh để xây dựng, đào tạo và triển khai mô hình machine learning một cách nhanh chóng.

- ***Phân tích theo thời gian thực***

Có thể dùng Redis kết hợp với các giải pháp phát trực tuyến như Apache Kafka và Amazon Kinesis làm kho dữ liệu trong bộ nhớ để tiêu thụ, xử lý và phân tích dữ liệu thời gian thực với độ trễ chưa đến một mili giây. Redis là lựa chọn lý tưởng cho các trường hợp sử dụng phân tích theo thời gian thực chẳng hạn như phân tích mạng xã hội, nhắm mục tiêu quảng cáo, cá nhân hóa và IoT.

1.6. Kết luận: Tại sao chúng ta sử dụng Redis?

Như đã giới thiệu ở trên, Redis hiện nay được cho là dễ dùng hơn một số loại key-value store khác vì nó vừa mang tính chất đơn giản và những ưu điểm đặc thù của một key-value store (mà điển hình là full-text search), lại vừa “thân thiện”, dễ dùng, đa tính năng nhờ vào việc có thể áp dụng những kiểu data types thông dụng như list, sort, set...

Redis là một sự lựa chọn tuyệt vời khi ta cần đến một server lưu trữ dữ liệu đòi hỏi tính mở rộng cao (scaleable) và chia sẻ bởi nhiều tiến trình, nhiều ứng dụng và nhiều server khác nhau. Chỉ riêng cơ chế tương tác giữa các tiến trình đã cực kỳ khó nhằn rồi. Do đó việc ta có thể tương tác cross-platform, cross-server, và cross-application đã làm Redis trở thành một lựa chọn đúng đắn cho rất rất nhiều công việc khác nhau. Tốc độ cực cao của Redis cũng có thể được lợi dụng để làm caching layer.

CHƯƠNG 2 - SO SÁNH VỚI SQL VÀ CÁC NoSQL KHÁC

2.1. SQL với Redis

Đặc trưng	Hệ thống quản lý cơ sở dữ liệu quan hệ (RDBMS)	Redis
Khối lượng công việc tối ưu	Truy vấn đặc biệt; kho dữ liệu; OLAP (xử lý phân tích trực tuyến).	Kho dữ liệu trong bộ nhớ, mã nguồn mở để sử dụng làm cơ sở dữ liệu, bộ nhớ đệm, trình chuyển tiếp tin nhắn và danh sách tác vụ chờ xử lý.
Mô hình dữ liệu	Mô hình quan hệ yêu cầu một lược đồ được xác định rõ, nơi dữ liệu được chuẩn hóa thành các bảng, các hàng và các cột. Ngoài ra, tất cả các mối quan hệ được xác định giữa các bảng, cột, chỉ mục và các phân tử cơ sở dữ liệu khác.	Redis lưu trữ data dưới dạng KEY-VALUE được lưu trữ trong RAM. Các kiểu dữ liệu Redis dùng để lưu value gồm: string, list, set, hash, Zset (sorted set)
Truy cập dữ liệu	SQL (Structured Query Language) là tiêu chuẩn để lưu trữ và truy xuất dữ liệu. Các cơ sở dữ liệu quan hệ cung cấp một bộ công cụ phong phú để đơn giản hóa việc phát triển các ứng dụng dựa trên cơ sở dữ liệu, nhưng tất cả các công cụ này đều sử dụng SQL.	Nó hỗ trợ nhiều cấu trúc dữ liệu như strings, hashes, lists, sets, sorted sets với truy vấn phạm vi, bitmaps, hyperloglogs, geospatial indexes với radius queries và streams, cùng với scripting bằng ngôn ngữ Lua.
Hiệu suất	Cơ sở dữ liệu quan hệ được tối ưu hóa để lưu trữ, do đó hiệu năng thường phụ thuộc vào hệ thống con đĩa. Nhà phát triển và quản trị viên cơ sở dữ liệu phải tối ưu hóa truy vấn, chỉ mục và cấu trúc bảng để đạt được hiệu suất cao nhất.	Toàn bộ dữ liệu Redis nằm trong bộ nhớ chính của máy chủ, trái với cơ sở dữ liệu, chẳng hạn như PostgreSQL, Cassandra, MongoDB, v.v. thường lưu phần lớn dữ liệu trên ổ đĩa hoặc ổ SSD. So với cơ sở dữ liệu trên ổ đĩa truyền thống trong đó phần lớn các tác vụ đều yêu cầu truy cập qua lại tới ổ đĩa, kho dữ liệu trong bộ nhớ chẳng hạn như Redis không phải chịu hình phạt này. Do đó kho dữ liệu kiểu này có thể hỗ trợ thêm được khá nhiều tác vụ và có thời gian phản hồi nhanh hơn. Kết quả là – hiệu suất nhanh thấy rõ với các tác

		vụ đọc hoặc ghi thông thường mất chưa đầy một mili giây và hỗ trợ hàng triệu tác vụ mỗi giây.
Khả năng mở rộng	Nó là dễ nhất để mở rộng với phần cứng nhanh hơn. Cũng có thể cho các bảng cơ sở dữ liệu trải rộng trên nhiều máy chủ trong một hệ thống phân tán, nhưng điều này đòi hỏi đầu tư bổ sung. Cơ sở dữ liệu quan hệ có kích thước tối đa cho số lượng và kích thước tệp, áp dụng giới hạn trên về khả năng mở rộng.	Redis là dự án mã nguồn mở được một cộng đồng đông đảo ủng hộ. Không có giới hạn về nhà cung cấp hoặc công nghệ vì Redis được có tính tiêu chuẩn mở, hỗ trợ các định dạng dữ liệu mở và có tập hợp máy khách phong phú.

2.2. Redis với NoSQL khác

Với NoSQL, dữ liệu có thể được lưu trữ theo kiểu đơn giản lược đồ hoặc dạng tự do. Dữ liệu bất kỳ có thể được lưu trữ trong bản ghi bất kỳ. Trong số các cơ sở dữ liệu NoSQL, có 4 mô hình lưu trữ dữ liệu phổ biến, do đó, có 4 loại hệ thống NoSQL phổ biến:

- Key-value stores (Redis, Riak...)
- Document database (MongoDB, CouchBD...)
- Wide column stores (Cassandra, Hbase...)
- Graph database (Neo4j...)

Giờ chúng ta sẽ so sánh các NoSQL này với nhau với **Redis** là đại diện của loại **Key-value stores** để có cái nhìn tổng quan về sự khác biệt giữa Redis (Key-value stores) với các NoSQL khác, qua đó biết cách để chọn loại mô hình lưu trữ dữ liệu phù hợp với dự án của mình.

	Key-value stores	Document database	Wide column stores	Graph database
Mô tả	Dữ liệu được lưu trữ trong database dưới dạng key-value, giống như một Dictionary trong C#. Để truy vấn dữ liệu trong database, ta dựa	Mỗi object sẽ được lưu trữ trong database dưới dạng một document. Dữ liệu sẽ được lưu trữ dưới dạng BSON/JSON/XML dưới database. Dữ	Dữ liệu được lưu trong database dưới dạng các cột, thay vì các hàng như SQL. Mỗi hàng sẽ có một key/id riêng. Điểm đặt biệt là	Dữ liệu được biểu diễn dưới dạng mạng hoặc đồ thị của các thực thể và các mối quan hệ của thực thể đó, với mỗi

	vào key để lấy value ra.	liệu không schema cứng như SQL, do đó ta có thể thêm/sửa field, thay đổi table, ... rất nhanh và đơn giản.	các hàng trong một bảng sẽ có số lượng cột khác nhau. Câu lệnh truy vấn của nó khá giống SQL.	node trong biểu đồ là một khối dữ liệu ở dạng tự do.
Ưu điểm	Tìm kiếm rất nhanh.	Dùng khi dữ liệu nguồn không được mô tả đầy đủ.	Tìm kiếm nhanh, Phân tán dữ liệu tốt.	Ứng dụng các thuật toán trên đồ thị như Đường đi ngắn nhất, liên thông,...
Nhược điểm	Lưu dữ liệu không theo khuôn dạng (schema) nhất định.	Hiệu năng truy vấn, Không có cú pháp chuẩn cho câu truy vấn dữ liệu.	Hỗ trợ được với rất ít phần mềm.	Phải duyệt nội bộ đồ thị, để trả lời lại các truy vấn. Không dễ để phân tán.
Ứng dụng	Do tốc độ truy xuất nhanh, key-value database thường được dùng để làm cache cho ứng dụng (Tiêu biểu là Redis và MemCache). Ngoài ra, nó còn được dùng để lưu thông tin trong sessions, profiles/preferences của user...	Do nhanh và linh động, document database thường đóng vai trò làm database cho các ứng dụng prototype, big data, e-commerce, CMS. Ngoài ra, ta còn dùng nó để lưu log hoặc history.	Column-Family Database được sử dụng khi ta cần ghi một số lượng lớn dữ liệu, big data. Nó còn được ứng dụng trong 1 số CMS và ứng dụng e-commerce.	Khi cần truy vấn các mối quan hệ, graph database truy vấn nhanh và dễ hơn nhiều so với database. Nó được dùng trong các hệ thống: mạng nơ ron, chuyên tiền bạc, mạng xã hội (tìm bạn bè), giới thiệu sản phẩm (dựa theo sở thích/ lịch sử mua sắm của người dùng)...

CHƯƠNG 3 - CÀI ĐẶT VÀ TRUY VẤN

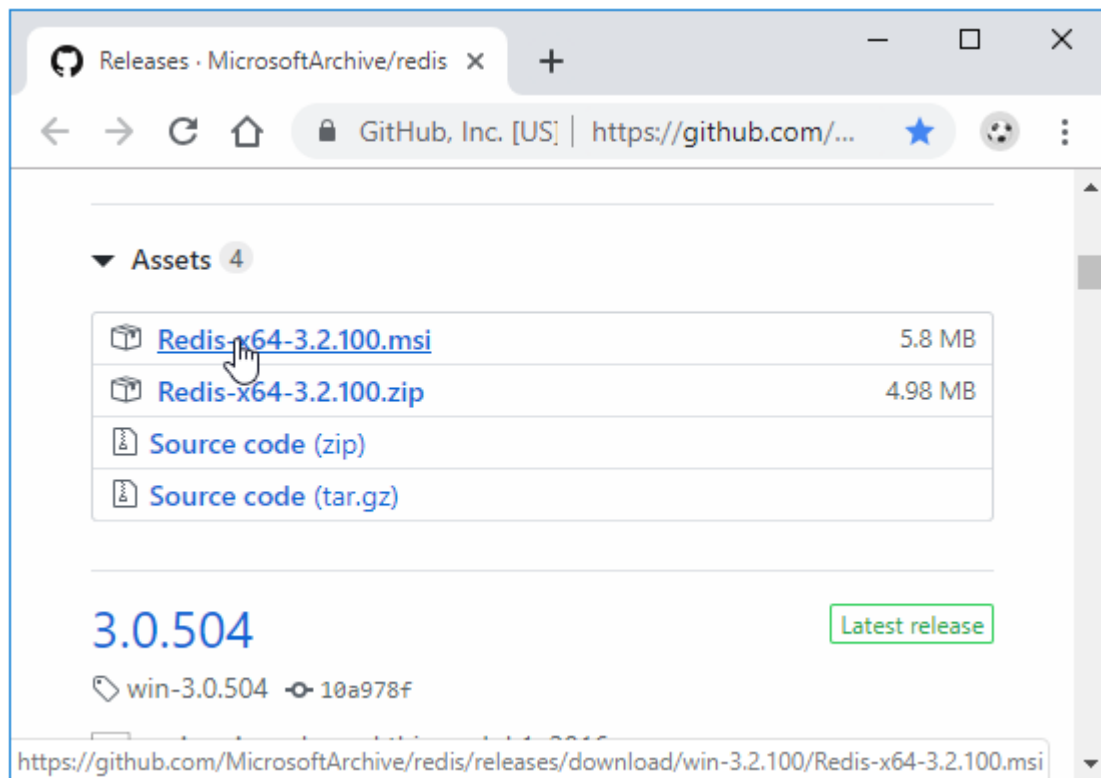
3.1. Cách cài đặt Redis-Server trên Windows

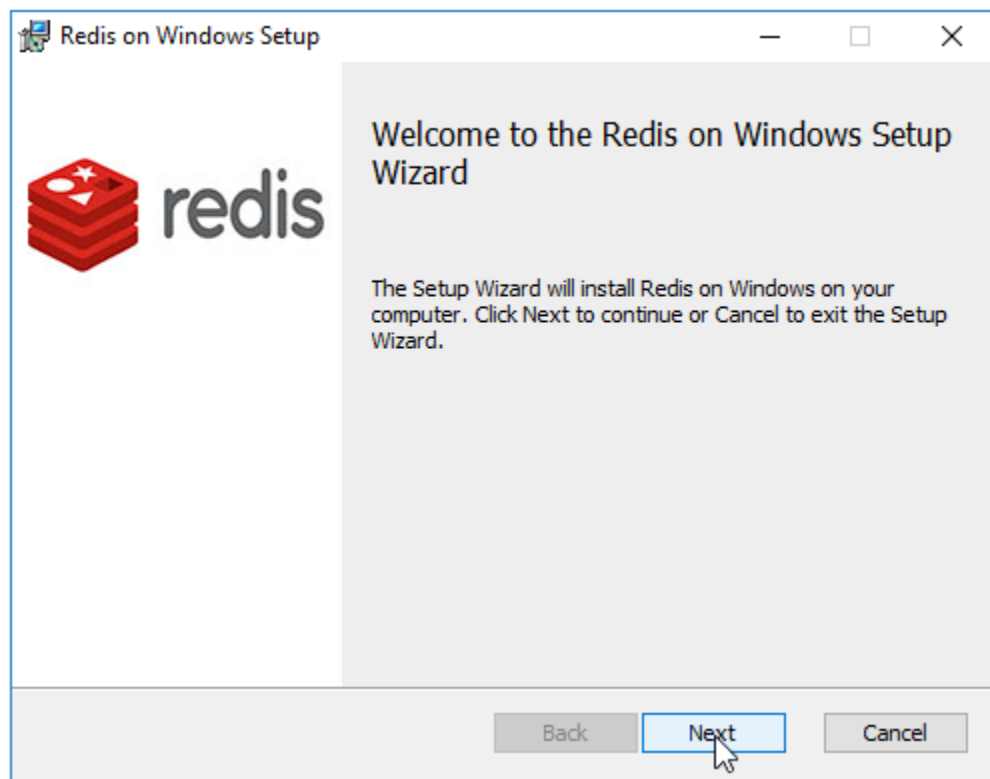
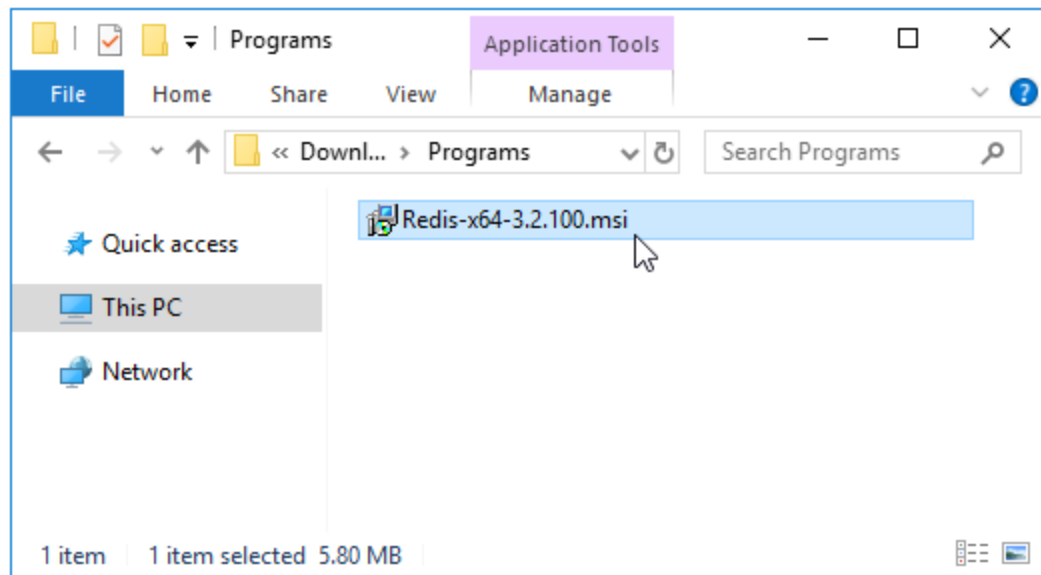
Bạn download các phiên bản tại:

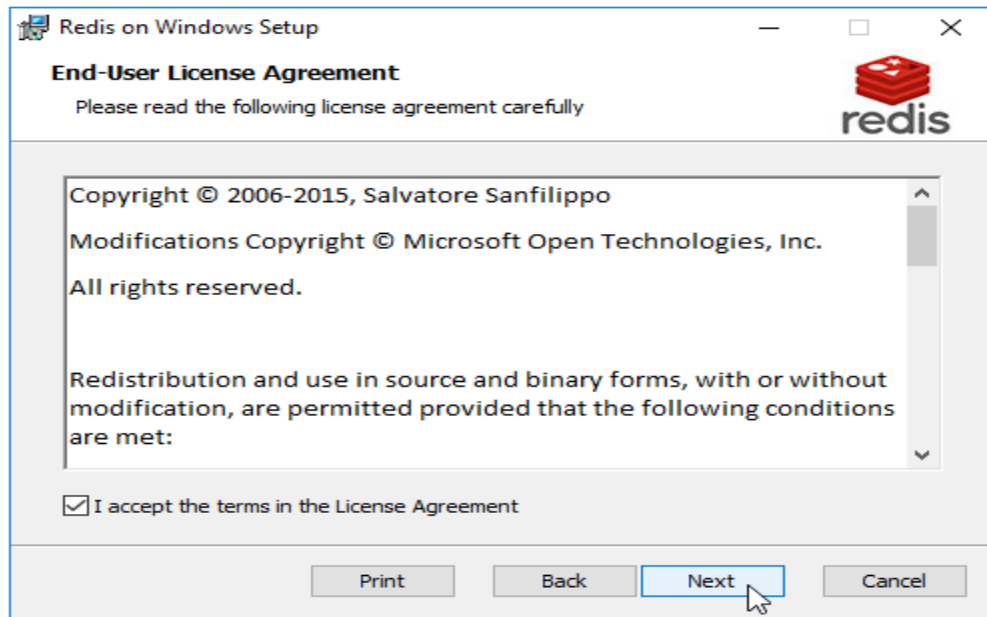
<https://github.com/MicrosoftArchive/redis/releases> để chạy trên windows

Bạn có thể download file cài đặt **.msi** hoặc file **.zip** (giải nén về chạy file **redis-server.exe** là được)

Ở đây mình tải file **.msi** để cài đặt. Trong quá trình cài đặt nó sẽ cho phép chúng ta tùy chỉnh một số tham số theo ý muốn.

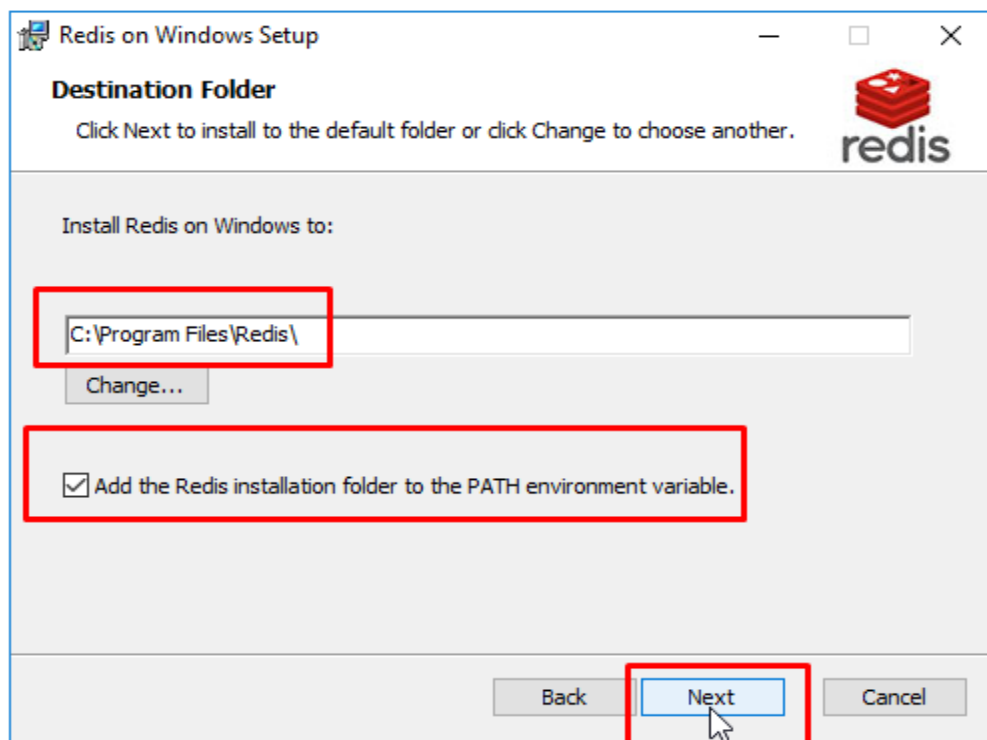




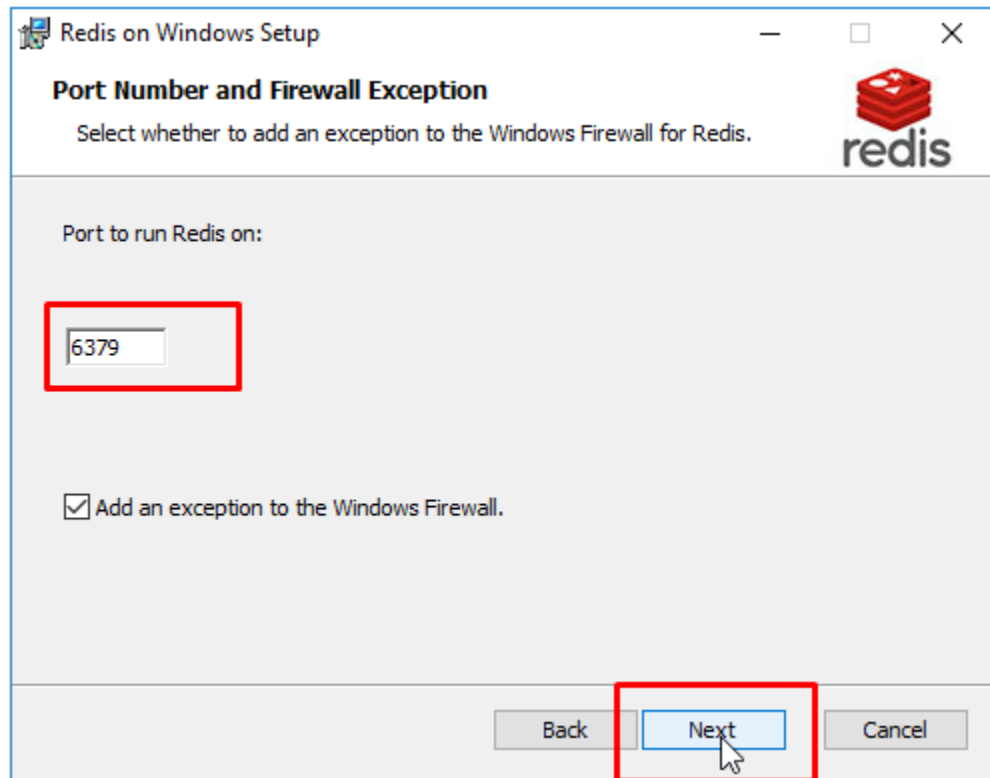


Trong phần ***Install Redis on Windows to*** các bạn chọn thư mục muốn cài đặt

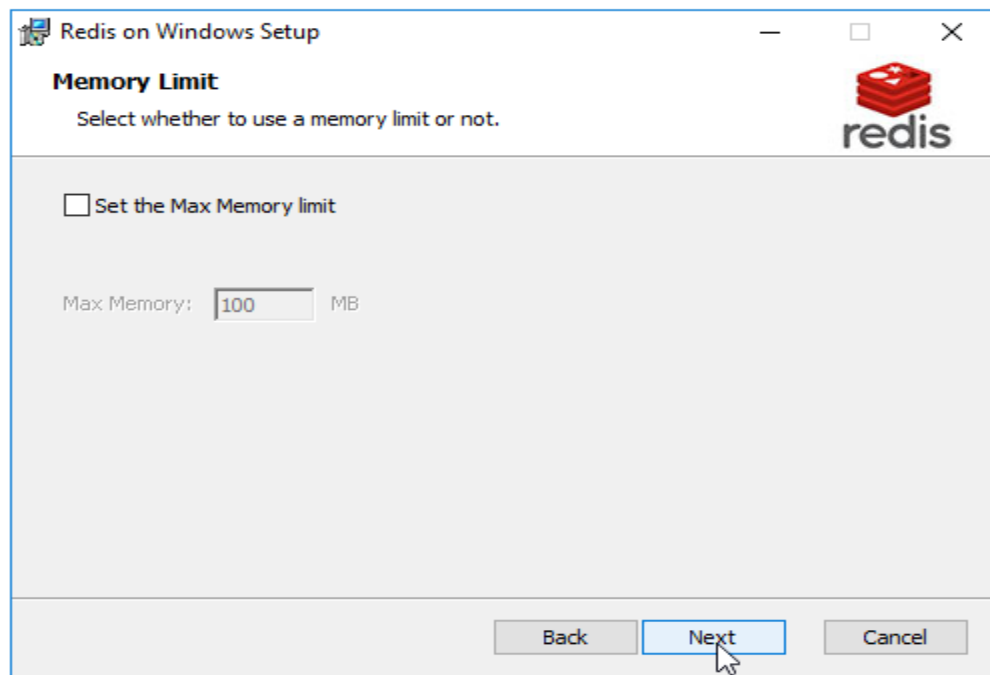
Nhớ chọn ô ***Add the Redis*** ... nó sẽ tự động thêm thư mục cài đặt vào biến môi trường, sau đó bạn có thể chạy lệnh redis-server, redis-cli ở bất kỳ thư mục nào trong màn hình cmd/powershell

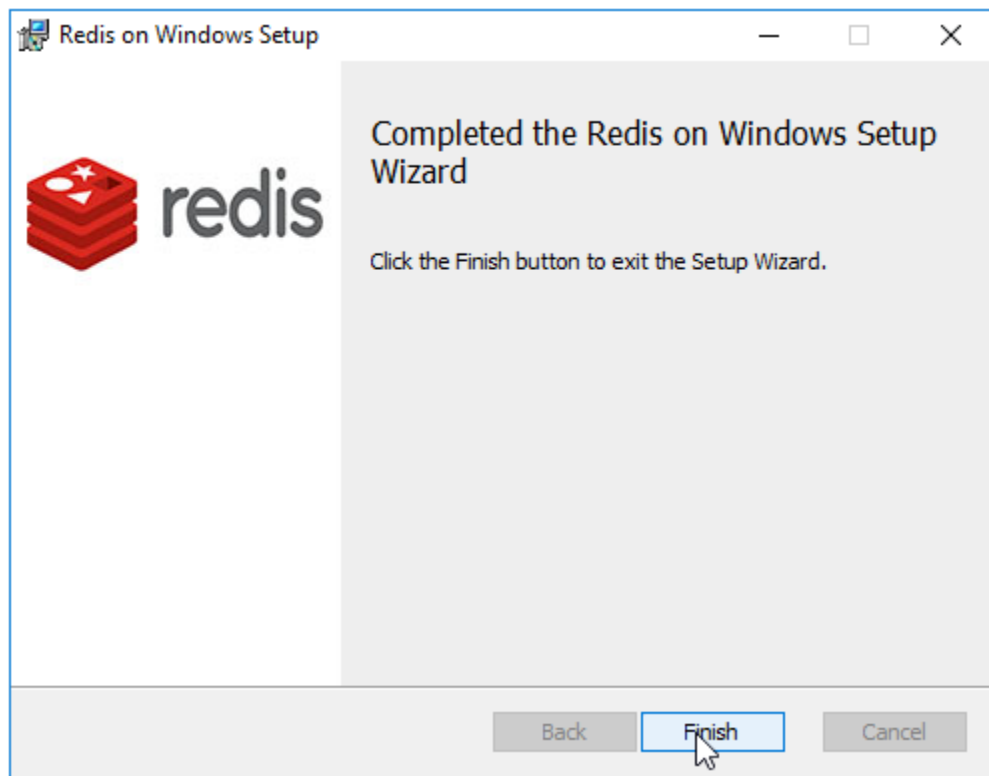
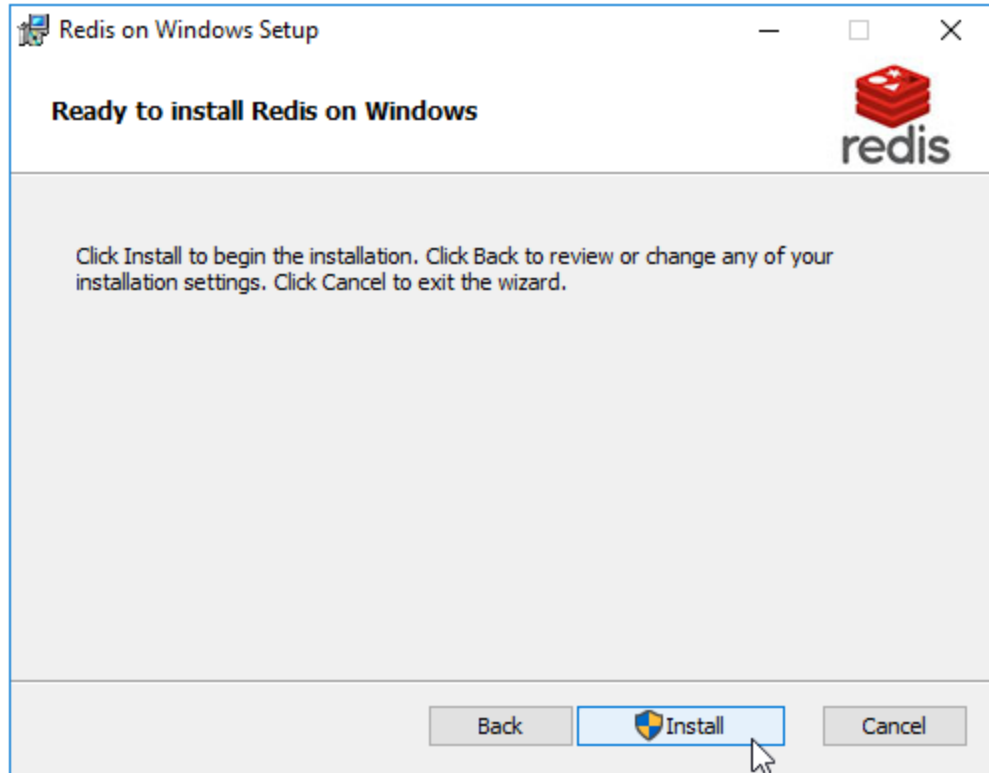


Chọn Port cho Redis Server (mặc định là 6379)

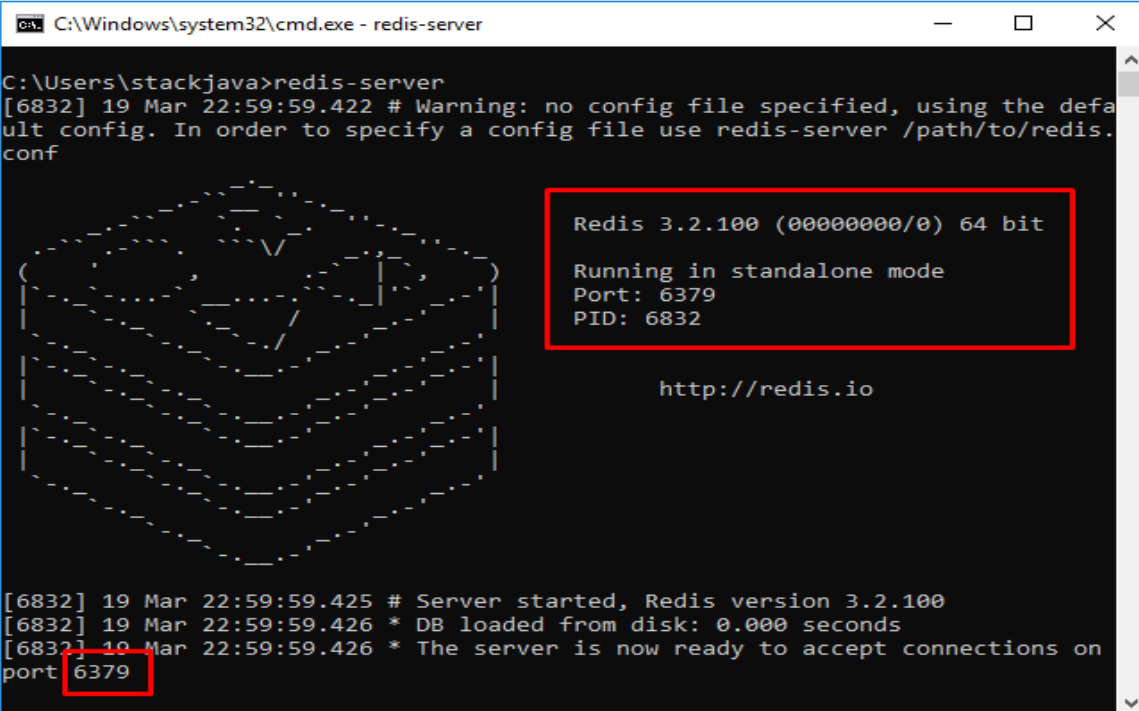


Bạn cũng có thể giới hạn dung lượng tối đa dành cho Redis Server (ở đây mình không đặt giới hạn)





Sau khi cài đặt xong, mở màn hình cmd hoặc powershell và chạy lệnh **redis-server**



```
C:\Windows\system32\cmd.exe - redis-server
C:\Users\stackjava>redis-server
[6832] 19 Mar 22:59:59.422 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf

Redis 3.2.100 (00000000/0) 64 bit
Running in standalone mode
Port: 6379
PID: 6832

http://redis.io

[6832] 19 Mar 22:59:59.425 # Server started, Redis version 3.2.100
[6832] 19 Mar 22:59:59.426 * DB loaded from disk: 0.000 seconds
[6832] 19 Mar 22:59:59.426 * The server is now ready to accept connections on port 6379
```

Để sử dụng file **redis.conf** cấu hình cho **redis-server** bạn có thể chạy lệnh **redis-server** với đường dẫn tới file **redis.conf**

Ví dụ **redis-server C:/redis.conf**

Chạy lệnh **redis-cli** để connect tới redis server và thực hiện thêm, lấy dữ liệu

set name kai: lưu chuỗi text kai vào biến name

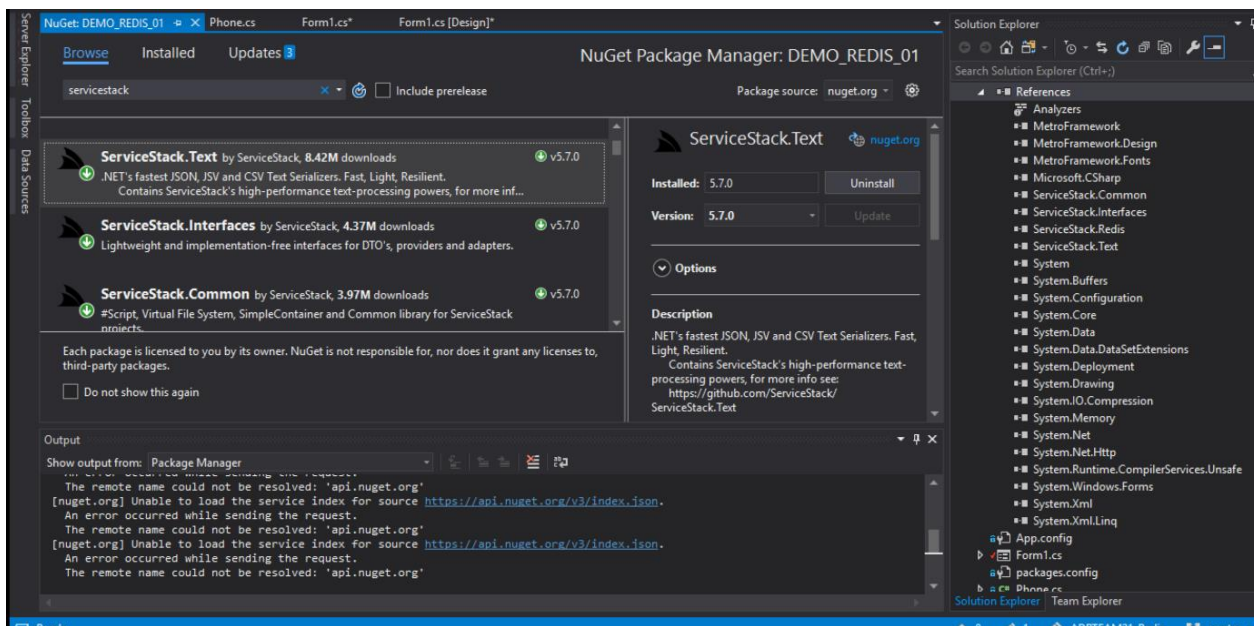
get name: lấy giá trị trong biến name

```
C:\Windows\system32\cmd.exe - redis-cli

C:\Users\stackjava>redis-cli
127.0.0.1:6379> set name kai
OK
127.0.0.1:6379> get name
"kai"
127.0.0.1:6379> _
```

3.2. Giới thiệu cách truy vấn Redis bằng C#

Bước đầu cài đặt package hỗ trợ kết nối Redis và C# Servicestack qua nudget:



Kết nối Redis bằng C#

```

1 reference
private void Form1_Load(object sender, EventArgs e)
{
    using (RedisClient client = new RedisClient("localhost", 6379))
    {
        IRedisTypedClient<Phone> phone = client.As<Phone>();
        phoneBindingSource.DataSource = phone.GetAll();
        Edit(true);
    }
}

```

Thêm dữ liệu

```

using (RedisClient client = new RedisClient("localhost", 6379))
{
    phoneBindingSource.EndEdit();
    IRedisTypedClient<Phone> phone = client.As<Phone>();
    List<Phone> ListWillStore = new List<Phone>();
    foreach (var temp in phoneBindingSource.DataSource as List<Phone>)
    {
        if (!temp.IsNull())
        {
            ListWillStore.Add(temp);
        }
    }
    phone.StoreAll(ListWillStore);
}

```

Xóa dữ liệu

```

Phone p = phoneBindingSource.Current as Phone;
using (RedisClient client = new RedisClient("localhost", 6379))
{
    IRedisTypedClient<Phone> phone = client.As<Phone>();
    phone.DeleteById(p.ID);
    phoneBindingSource.RemoveCurrent();
    ClearText();
    UpdateDB();
}

```

Tìm kiếm dữ liệu

```

1 reference
private void BtnSearch_Click(object sender, EventArgs e)
{
    string Keyword = string.Empty;
    Keyword = metroTextBox1.Text;
    List<int> vs = new List<int>();
    using (RedisClient client = new RedisClient("localhost", 6379))
    {
        IRedisTypedClient<Phone> phone = client.As<Phone>();
        IList<Phone> phones = phone.GetAll();
        for (int i = 0; i < phones.Count; i++)
        {
            if (!phones[i].Model.Contains(Keyword))
            {
                phones.RemoveAt(i);
                i--;
            }
        }
        phoneBindingSource.DataSource = phones;
    }
}

```

Đổ dữ liệu vào datagrid view

```
1 reference
private void Form1_Load(object sender, EventArgs e)
{
    using (RedisClient client = new RedisClient("localhost", 6379))
    {
        IRedisTypedClient<Phone> phone = client.As<Phone>();
        phoneBindingSource.DataSource = phone.GetAll();
        Edit(true);
    }
}
```

Tài liệu tham khảo

1. <https://aws.amazon.com/vi/redis/>
2. <https://viblo.asia/p/gioi-thieu-ve-redis-aWj5382pK6m>
3. <https://viblo.asia/p/tong-quan-ve-redis-NznmMdXzMr69>
4. <https://medium.com/@swdream/redis-ph%E1%BA%A7n-1-redis-l%C3%A0-g%C3%AC-m%C3%A0-n%C3%B3-l%E1%BA%A1i-ph%E1%BB%95-bi%E1%BA%BF-n-v%C3%A0-quan-tr%E1%BB%8Dng-nh%C6%B0-th%E1%BA%BF-d0468a4a0d1e>
5. <https://kipalog.com/posts/Tim-hieu-Redis--Phan-1>
6. <https://db-engines.com/en/system/Microsoft+SQL+Server%3BRedis>
7. <https://techinsight.com.vn/nosql-co-gi-hay-phan-2-cac-dang-nosql-va-ung-dung/>
8. <https://kipalog.com/posts/Hieu-va-chon-database-trong-NoSQL-DBMS--Part1----Cac-loai-NoSQL-DBMS>