Link : “ http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server ”

**» Khắc phục sự cố máy chủ trực tiếp**

Hướng dẫn này cung cấp tổng quan về các chiến lược chính để phát hiện và báo cáo các sự cố về hiệu suất của máy chủ thời gian chạy. Chúng tôi sẽ mô tả một số vấn đề phổ biến như thiếu tài nguyên phần cứng, vấn đề phân luồng và rò rỉ bộ nhớ, đồng thời thảo luận các chiến lược để xác định nguyên nhân cơ bản.

* [**Cài đặt và sử dụng VisualVM**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#vvm)
* [**Phát hiện các vấn đề về bộ nhớ**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#mem)
* [**Phân tích hiệu suất CPU**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#cpu)
* [**Hiệu suất mạng**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#net)
* [**Soạn thảo và gửi yêu cầu hỗ trợ**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#support)

**LƯU Ý**   
Tài liệu này yêu cầu kiến thức cơ bản về hoạt động bên trong của Máy ảo Java (JVM), bao gồm quản lý bộ nhớ, tải lớp, phân luồng và thu gom rác.

**» Cài đặt và sử dụng VisualVM**

Đối với tất cả các ví dụ khắc phục sự cố của chúng tôi, chúng tôi sẽ làm việc với một công cụ miễn phí có tên là VisualVM do Oracle cung cấp trong mọi JDK (Bộ công cụ phát triển Java) gần đây, công cụ này cũng có sẵn dưới dạng bản [**tải xuống riêng từ trang web này**](http://visualvm.java.net/) .

Nếu bạn đã cài đặt JDK (phiên bản 6 hoặc cao hơn), bạn sẽ tìm thấy công cụ này trong thư mục **bin/** . Tệp thực thi được gọi là **jvisualvm** .

Công cụ này chạy trên mọi hệ điều hành chính. Nếu bạn chưa cài đặt nó trên máy của mình thì đây là thời điểm tốt để tải xuống và cài đặt nó.

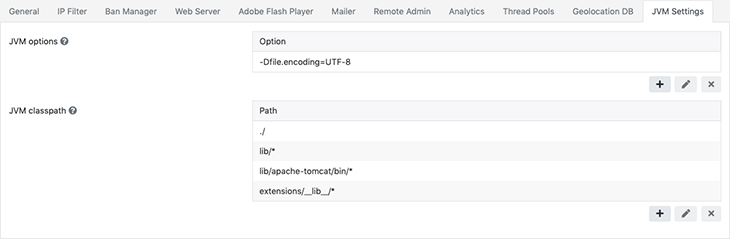
**» Làm việc với máy chủ từ xa**

VisualVM có thể cắm vào bất kỳ JVM cục bộ đang chạy nào chỉ bằng một cú nhấp chuột nhưng trong trường hợp máy chủ từ xa, chúng tôi cần định cấu hình JVM từ xa để có thể giao tiếp với công cụ qua mạng.

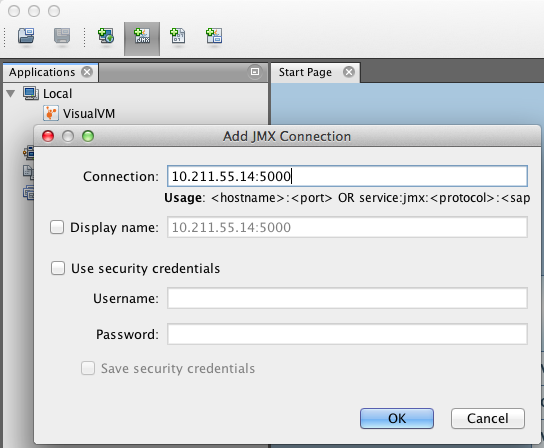
Để thực hiện việc này, chúng tôi sẽ mở Công cụ quản trị của SmartFoxServer, chọn mô-đun Trình **cấu hình máy chủ , nhấp vào tab Cài đặt JVM** và thêm các tham số sau vào phần **Tùy chọn JVM** :

-Dcom.sun.manager.jmxremote -Dcom.sun.man   
Quản lý.jmxremote.port=5000-Dcom.sun.man Quản lý.jmxremote.rmi.port=5000-Dcom.sun.man Quản lý.jmxremote.rmi.port=5000   
-Djava .rmi.server.hostname=[external-ip-address]   
-Dcom.sun.man Quản lý.jmxremote.authenticate=false-Dcom.sun.man Quản lý.jmxremote.ssl=false

Tham số **jmxremote.port** có thể là bất kỳ cổng TCP hợp lệ nào hoạt động tốt với máy chủ của bạn và không bị chặn bởi các quy tắc tường lửa.

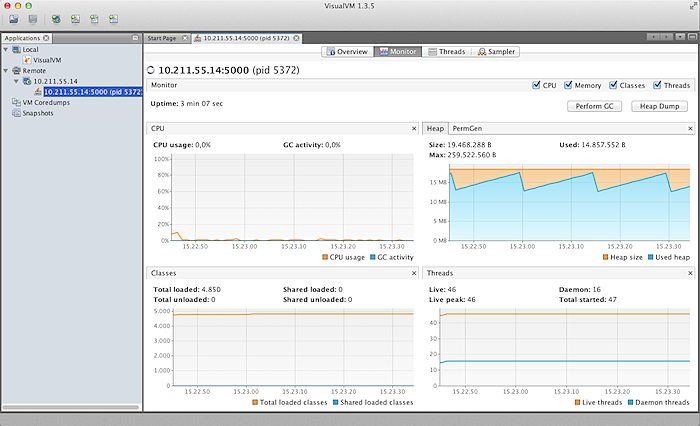


Sau khi bước này hoàn tất, chúng tôi có thể khởi động lại SFS và chúng tôi đã sẵn sàng kết nối qua VisualVM. Khởi chạy ứng dụng và nhấp vào biểu tượng **Thêm kết nối JMX** để bắt đầu kết nối từ xa:



Nhập địa chỉ IP từ xa (hoặc tên miền) của máy chủ và tên cổng ở dạng <hostname>:<port> và nhấp vào OK. Sau đó bấm đúp vào biểu tượng mới xuất hiện trên thanh bên trái.

Đây là giao diện của tab **Màn hình** khi được kết nối với máy chủ từ xa.



[**^ trên cùng**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#top)

**» Phát hiện các vấn đề về bộ nhớ**

Các vấn đề về bộ nhớ thấp trong Java thường khiến CPU tăng đột biến vì quy trình thu gom rác (GC) phải làm việc nhiều hơn. Có hai loại vấn đề chính liên quan đến quản lý bộ nhớ:

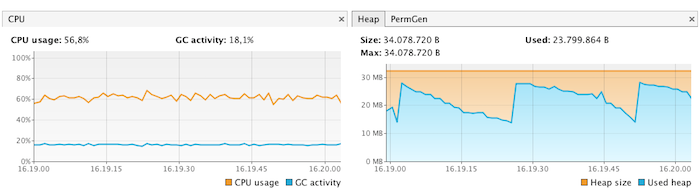
* Không đủ bộ nhớ để đáp ứng nhu cầu của ứng dụng
* Rò rỉ bộ nhớ

Trong trường hợp trước đây, chúng tôi thường tìm thấy một hoạt động liên tục của GC sẽ ảnh hưởng đến hiệu suất ứng dụng tổng thể ngay cả khi không có sự cố JVM.

Thay vào đó, cái sau sẽ liên tục tiêu thụ tài nguyên bộ nhớ cho đến khi JVM bị lỗi, với OutOfMemoryError.

**» Vấn đề bộ nhớ thấp**

Đây là một ví dụ về một ứng dụng đang chạy với bộ nhớ heap không đủ:

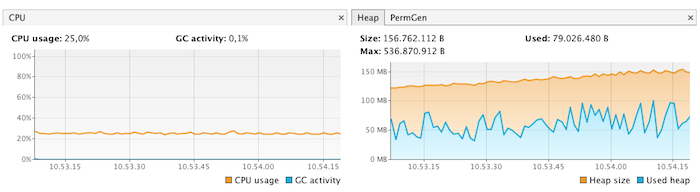


Vì thu gom rác là một quá trình không xác định, nên chỉ riêng biểu đồ bộ nhớ (bên phải) không thực sự cho chúng ta biết liệu chúng ta có đang thiếu bộ nhớ heap hay không. Tất cả những gì chúng ta có thể thấy là đã có một số chu kỳ phân bổ (đỉnh xanh) và một số lần phân bổ tiếp theo (đáy).

Chúng tôi thực sự có thể nhận thấy rằng giá trị **Tối đa** của bộ nhớ khả dụng là ~34 MB và các mức cao nhất gần như chạm mốc 30 MB, vì vậy đây là manh mối cho thấy cuối cùng chúng tôi có thể đạt đến mức cao nhất.

Điều thực sự quan trọng cần lưu ý là biểu đồ CPU/GC bên trái trong đó đường màu xanh lam hiển thị hoạt động liên tục của GC trong một thời gian khá dài, chiếm gần 1/3 CPU. Hoạt động này diễn ra càng lâu thì bằng chứng cho thấy bộ nhớ liên tục được dọn dẹp để tạo không gian cho các đối tượng mới càng rõ ràng.

Trong trường hợp này, việc tăng kích thước tối đa của heap từ 32MB lên 512MB sẽ giải quyết được vấn đề, như minh họa ở đây:



Hiện mức tiêu thụ CPU đã giảm đáng kể, hoạt động GC không đáng kể và mức sử dụng bộ nhớ tổng thể đã được cải thiện.

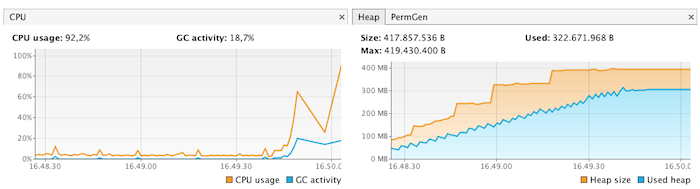
**" Rò rỉ bộ nhớ**

Trong Java, rò rỉ bộ nhớ xảy ra khi các đối tượng trong bộ nhớ không được giải phóng mặc dù bản thân ứng dụng không còn cần chúng nữa. Một ví dụ phổ biến là các trình xử lý sự kiện không được sử dụng chưa bị xóa khỏi nguồn sự kiện của chúng.

Nếu chương trình tiếp tục thêm các đối tượng nghe mới nhưng không bao giờ loại bỏ những đối tượng không còn được sử dụng, chúng ta sẽ có khả năng lãng phí bộ nhớ rất nhiều. GC sẽ không thể lấy lại bộ nhớ như vậy vì các trình nghe không được sử dụng vẫn được tham chiếu. Nếu các đối tượng này tiếp tục chồng chất lên nhau, chúng ta sẽ thấy sự suy giảm hiệu suất tăng dần, điều này có thể dẫn đến sự cố JVM.

Rò rỉ bộ nhớ không phải lúc nào cũng dễ dàng phát hiện ra, ẩn trong mã một thời gian trước khi chúng được phát hiện. Trong các trường hợp khác, rò rỉ có thể trở nên rất khó chịu rất nhanh chóng gây ra những đột biến lớn trong việc sử dụng bộ nhớ và cuối cùng là cái chết của tiến trình.

Đây là một ví dụ về rò rỉ bộ nhớ khó chịu trông như thế nào:

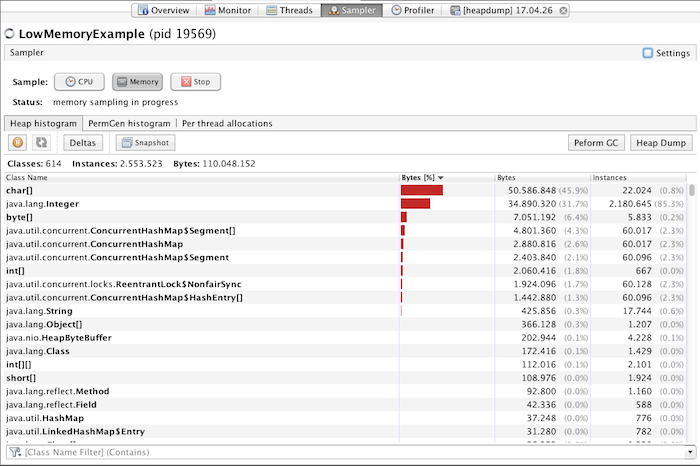


Xu hướng trong hoạt động bộ nhớ chỉ ra rằng GC không thể gọi lại không gian heap chưa sử dụng, do đó không ngừng tăng lên đến kích thước tối đa cho phép. Thông thường, loại biểu đồ này kết thúc bằng sự cố JVM, thường là do Lỗi OutOfMemory. Chúng ta có thể thấy rằng xung đột của CPU gần như chạm trần và GC đang hoạt động rất nhiều.

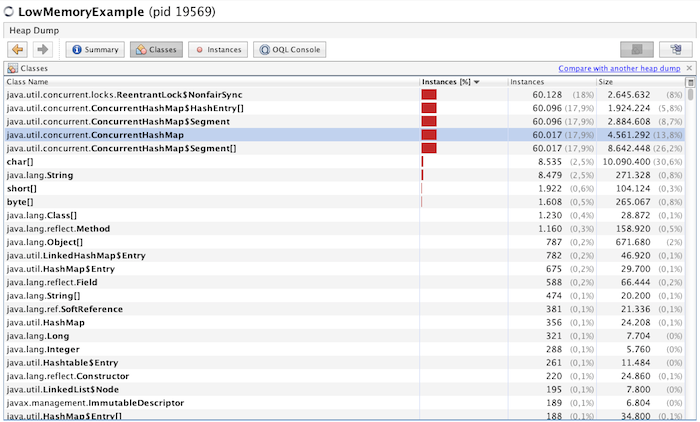
**»Xác định các nguồn sử dụng bộ nhớ quá mức**

Với **Memory Sampler** do VisualVM cung cấp, chúng ta có thể tìm hiểu sâu hơn về không gian heap JVM và kiểm tra xem lớp nào đang thực sự chiếm nhiều bộ nhớ. Chúng tôi có thể bắt đầu phiên lấy mẫu bằng cách nhấp vào tab Trình lấy mẫu và nhấn **nút Bộ nhớ** .

Chế độ xem trực tiếp trông như thế này:



Sau đó, chúng ta có thể nhấn nút **Heap Dump** để có ảnh chụp nhanh về phân bổ bộ nhớ:



Kết xuất cho thấy rõ ràng rằng hầu hết các đối tượng sử dụng nhiều bộ nhớ là các thể hiện của đối tượng ConcurrentHashMap. Sau đó, chúng ta có thể đào sâu hơn trong lớp để phân tích từng trường hợp và xem nội dung của chúng. Khi chúng tôi biết vấn đề là gì, chúng tôi có thể quay lại mã của mình và điều tra lý do rò rỉ.

Thông thường, một kết xuất bộ nhớ sẽ đủ để cung cấp manh mối về bản chất của vấn đề. Trong trường hợp điều này là không đủ, người ta có thể cần truy cập vào mức độ chi tiết tốt hơn bằng Trình lập hồ sơ Java chuyên dụng.

**» Lỗi PermGen**

Ngoài ra còn có một loại lỗi bộ nhớ khác liên quan đến phần [**Thế hệ vĩnh viễn**](https://blogs.oracle.com/jonthecollector/entry/presenting_the_permanent_generation) (viết tắt là PermGen) của bộ nhớ JVM. PermGen là vùng bộ nhớ dành riêng cho các định nghĩa lớp. Nói cách khác, đó là phần của bộ nhớ JVM, nơi tất cả dữ liệu từ các lớp được tải của chúng tôi đi vào.

Khi làm việc với SmartFoxServer 2X, có thể bão hòa bộ nhớ PermGen và gây ra lỗi đáng sợ **OutOfMemoryError: PermGen space** error.

Nguyên nhân của vấn đề này thường là do tải lại Tiện ích mở rộng liên tục, từ đó tạo ra các bản sao mới của các lớp Tiện ích mở rộng, có thể làm cạn kiệt bộ nhớ PermGen có sẵn.

Chúng tôi cung cấp mô tả chi tiết về cách tải lớp hoạt động với SmartFoxServer và các cách để tránh sự cố [**này trong bài viết này**](http://docs2x.smartfoxserver.com/ExtensionsJava/advanced-concepts#classLoading) .

Trong VisualVM bên dưới phần **Màn hình** , chúng ta có thể chuyển đổi giữa chế độ xem bộ nhớ **Heap và chế độ xem bộ nhớ PermGen** để theo dõi cả hai vùng bộ nhớ.

**LƯU Ý**   
Trong Java 8 và bộ nhớ **PermGen cao hơn đã được thay thế bằng cái gọi là Meta-Space** . Mặc dù có một số khác biệt về kỹ thuật trong cách chúng hoạt động bên trong, nhưng về cơ bản chúng có chức năng giống nhau. Bạn có thể tìm hiểu thêm về nó [**trong bài viết bên ngoài này**](http://java.dzone.com/articles/java-8-permgen-metaspace) .

[**^ trên cùng**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#top)

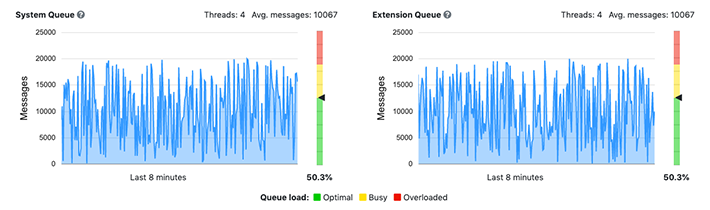
**» Phân tích hiệu suất CPU**

Để kiểm tra việc sử dụng CPU của SmartFoxServer trực tiếp, bạn có thể sử dụng một số công cụ:

* Trình **giám sát tài nguyên riêng của hệ điều hành** sẽ hiển thị cho bạn mức sử dụng CPU toàn cầu và phân phối tải trên nhiều bộ xử lý lõi
* **Công cụ quản trị của SmartFoxServer** cung cấp thêm chi tiết về luồng nào đang hoạt động mạnh hơn và trạng thái của hàng đợi nội bộ.
* **VisualVM Monitor** và **Sampler** để kiểm tra chi tiết về việc sử dụng tài nguyên JVM.

Nếu máy chủ không phản hồi nhanh trong khi tài nguyên CPU gần như miễn phí, bạn có thể nghi ngờ rằng một số luồng đang lãng phí thời gian chờ các tài nguyên khác khả dụng. Đây là điển hình của các truy vấn cơ sở dữ liệu hoặc các cuộc gọi chặn khác tới các tài nguyên bên ngoài (dịch vụ web, cuộc gọi http từ xa, v.v.)

Trong trường hợp này, hãy xem hàng đợi của Máy chủ có thể sẽ cho biết vấn đề nằm ở đâu. Ví dụ dưới đây cho thấy một tình huống trong đó máy chủ đang cố gắng theo kịp tải hiện tại. Do số lượng luồng tương đối thấp được sử dụng bởi cả hai bộ điều khiển, vấn đề có thể sẽ biến mất bằng cách tăng chúng cho cả Hệ thống và Tiện ích mở rộng.



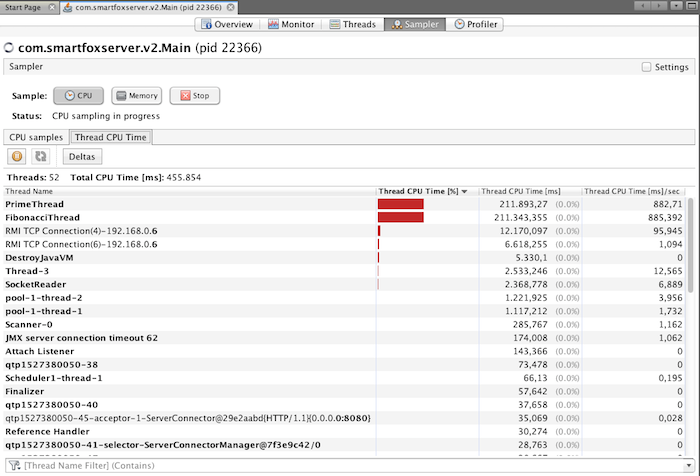
**LƯU Ý**   
Cấu hình sai luồng có nhiều khả năng xảy ra với các phiên bản SmartFoxServer trước 2.9.0 trong đó nhóm luồng được cấu hình thủ công. Kể từ phiên bản 2.9.0, chúng tôi đã giới thiệu nhóm luồng tự động điều chỉnh tỷ lệ để tránh cấu hình thủ công và không cần bảo trì nhỏ hoặc không cần bảo trì.

**» Xác định nguồn tiêu thụ CPU**

Bằng cách chọn chế độ xem Trình lấy **mẫu** trong VisualVM và nhấp vào **nút CPU,** chúng ta có thể tìm hiểu sâu hơn về JVM và khám phá lớp nào đang làm việc chăm chỉ hơn và chiếm hầu hết các chu kỳ CPU.

Đối với ví dụ này, chúng tôi đã khởi chạy một Tiện ích mở rộng đơn giản, lần lượt bắt đầu hai luồng có tên là *PrimeRunner* và *FibonacciRunner* , sẽ thực hiện nhiều phép tính, tạo ra càng nhiều số nguyên tố và giá trị fibonacci càng tốt.

Bằng cách chuyển sang chế độ xem **Thời gian CPU của luồng** , chúng ta có thể thấy ngay tác động của thử nghiệm này:



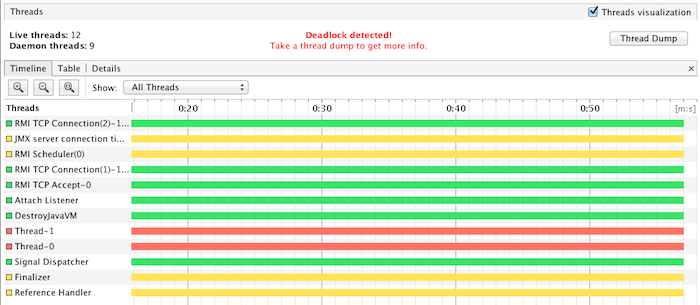
Hai luồng làm việc chăm chỉ nằm ở đầu danh sách, hiển thị rõ ràng ai đang sử dụng hầu hết các chu kỳ CPU. Chúng tôi cũng có thể tiến hành lấy toàn bộ kết xuất luồng bằng cách quay lại chế độ xem **mẫu CPU** và nhấp vào nút **Kết xuất luồng** . Điều này có thể hữu ích để xem phương thức nào hiện đang được thực thi trong mỗi luồng.

**LƯU Ý:**   
Hãy nhớ rằng việc lấy mẫu CPU không kỹ lưỡng bằng việc lập hồ sơ ứng dụng. Trình lược tả có thể tạo ra số liệu thống kê hiệu suất chi tiết hơn nhiều, nhưng thật không may, đó là một quy trình sử dụng nhiều tài nguyên hơn và không thể chạy trên máy chủ trực tiếp.   
  
Nếu phát hiện một vấn đề khó khăn về hiệu suất trong mã của mình, bạn có thể cần chạy VisualVM Profiler (hoặc trình lược tả bên thứ 3) trong môi trường thử nghiệm cục bộ của mình để tìm hiểu sâu hơn về vấn đề.

**» Chủ đề bế tắc**

Một vấn đề cuối cùng có thể khiến máy chủ không phản hồi là [**bế tắc luồng**](http://en.wikipedia.org/wiki/Deadlock) . Đây là một tình huống đáng sợ có thể xảy ra khi hai hoặc nhiều luồng đang chờ kết thúc một hành động cạnh tranh và chờ lẫn nhau, gây ra khối vô thời hạn.

Một lần nữa VisualVM sẽ đến giải cứu với chế độ xem **Chủ đề của anh ấy** , làm nổi bật các bế tắc cho chúng tôi:



Hai luồng được đánh dấu màu đỏ dường như đang tranh giành cùng một tài nguyên, chờ đợi nhau vô thời hạn. Bước tiếp theo là lấy kết xuất chuỗi bằng cách nhấp vào nút trên cùng bên trái và xem chính xác vị trí chính xác trong mã mà các chuỗi đó bị kẹt.

[**^ trên cùng**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#top)

**» Sự cố mạng**

Các sự cố liên quan đến mạng có thể khó xác định hơn vì chúng nằm ngoài các công cụ giám sát mà chúng tôi đã mô tả ở đây. VisualVM nói riêng sẽ không giúp được gì nhiều về vấn đề này, trong khi AdminTool có thể được sử dụng để tìm kiếm manh mối về độ trễ hoặc tốc độ mạng quá mức.

Chúng tôi có hẳn một bài viết cụ thể về vấn đề mạng mà [**bạn có thể tham khảo tại đây**](http://docs2x.smartfoxserver.com/GettingStarted/troubleshooting) .

[**^ trên cùng**](http://docs2x.smartfoxserver.com/AdvancedTopics/troubleshooting-live-server#top)

**» Soạn và gửi yêu cầu hỗ trợ**

Thông thường, chúng tôi nhận được các câu hỏi về vấn đề điều chỉnh hiệu suất hoặc tài nguyên thiếu các chi tiết cơ bản mà chúng tôi yêu cầu để trợ giúp.

Trước khi liên hệ với bộ phận hỗ trợ của chúng tôi, vui lòng đảm bảo rằng bạn đã thu thập các thông tin sau:

* Phiên bản SmartFoxServer
* Loại hệ điều hành, nhà cung cấp và phiên bản
* Tóm tắt cơ bản về phần cứng của máy chủ (loại CPU/lõi/tốc độ, RAM, băng thông mạng)
* Nền tảng ứng dụng khách (Unity, iOS, Android, v.v.) và số phiên bản của API ứng dụng khách
* Mô tả tổng hợp của vấn đề
* Chỉ định xem sự cố xảy ra không liên tục hay xảy ra mọi lúc. Nếu là trường hợp sau, hãy cung cấp mô tả từng bước về cách tái tạo sự cố (nếu có thể)

Tài liệu bổ sung:

* Phiên bản nén của thư mục SFS2X/config/
* Kiểm tra nhật ký phía máy chủ (SFS2X/logs/smartfox.log) để tìm lỗi và báo cáo mọi cảnh báo hoặc lỗi có thể liên quan đến sự cố được đề cập. Nếu bạn cần gửi tệp nhật ký, hãy nhớ nén chúng.
* Nếu sự cố được phát hiện thông qua AdminTool, vui lòng đính kèm ảnh chụp màn hình có liên quan cùng với báo cáo.
* Nếu sự cố được phát hiện thông qua VisualVM, vui lòng đính kèm ảnh chụp màn hình, kết xuất luồng hoặc ảnh chụp nhanh tài nguyên vào yêu cầu hỗ trợ

Yêu cầu hỗ trợ có thể được gửi đến hộp thư điện tử **support@** ... của chúng tôi.