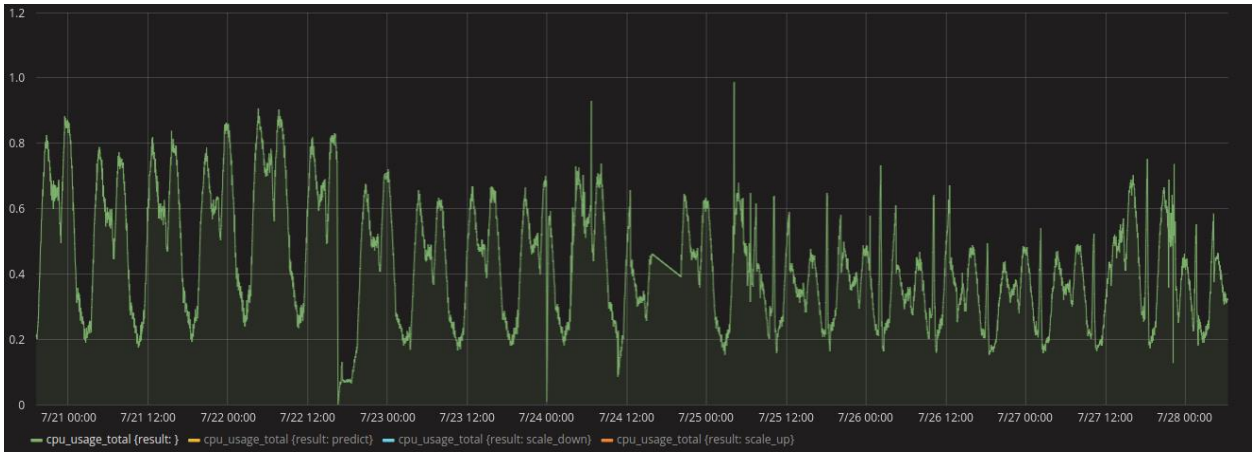
Kết quả thử nghiệm

1. **Môi trường thử nghiệm.**
   - Chạy hệ thống scale trên Ubuntu Server 16.04
   - Chạy Openstack Newton trên Ubuntu Server 16.04
   - Công cụ jmeter để bắn request giả lập người dùng.
   - Máy ảo đám mây:
     - Cloud image: Ubuntu Server 16.04, flavor 1 vCPU, 1GB Ram, 5GB ổ cứng.
     - Cài đặt app server python bằng thư viện falcon đơn giản trên các máy ảo đám mây, sử dụng web server gunicorn. Server này nhận và sử lý request từ jmeter.
     - Sử dụng cadvisor, influxdb để theo dõi máy ảo.
2. **Dữ liệu thử nghiệm.**
   - Sử dụng jmeter bắn request tới gunicorn server trên máy ảo, tài nguyên tiêu thụ của máy ảo trong quá trình thử nghiệm được theo dõi bởi cadvisor với backend là influxdb theo thời gian thực.
   - Dữ liệu thử nghiệm là dữ liệu phần trăm cpu tiêu thụ trên máy ảo, được lấy từ cadvisor/influxdb cài trên máy ảo, tính trung bình theo đơn vị phút, lưu vào trong influxdb.
   - Sinh kịch bản số lượng request theo thời gian có chu kỳ 8 giờ. Các request được bắn bằng jmeter theo kịch bản này.



Hình: Dữ liệu tài nguyên tiêu thụ của một máy ảo đám mây khi bắn request thử nghiệm bằng jmeter, lấy từ cadvisor/influxdb, tính trung bình mỗi 2 phút. Đồ thị được hiển thị bằng grafana.

3. **Mục đích của thử nghiệm.**
   Thử nghiệm độ hiệu quả và khả năng dự đoán của phương pháp dự đoán PD GABP trên dữ liệu thời gian thực.

- So sánh các mô hình dự đoán (2-4: lấy mẫu dữ liệu trung bình và dự đoán 2 phút 1 lần, dự đoán trước 4 phút tương lai, hoặc 4-4: lấy mẫu dữ liệu trung bình và dự đoán 4 phút 1 lần, dự đoán trước 4 phút tương lai)
- Độ hiệu quả khi tài nguyên tiêu thụ thay đổi đột ngột.
- Độ hiệu quả khi tài nguyên tiêu thụ không có đột biến.

4. Thiết lập chung.

Cho phép tối đa 2 máy ảo đám mây một lúc (1 theo dõi, 1 giãn thêm, máy ảo giãn thêm không được theo dõi).

Trong khuôn khổ thử nghiệm xoay quan phần trăm CPU tiêu thụ. Thử nghiệm quy định nếu phần trăm CPU lớn hơn 55% được coi là quá tải (tương đương 100% trong thực tế).

Thiết lập luật đơn giản để hủy bớt máy ảo khi tài nguyên tiêu thụ thấp: nếu phần trăm CPU nhỏ hơn 20% thì giảm bớt 1 máy ảo.

Mạng neural được huấn luyện lại mỗi 8 tiếng một lần. Jmeter giả lập số lượng request theo chu kỳ 8 tiếng.

Độ dài dữ liệu được huấn luyện là khoảng 4 ngày.

Phương pháp PDGABP, tham số windowsize = 4, m = 1.

5. Thử nghiệm.

5.1. So sánh các mô hình dự đoán khi dữ liệu thay đổi đột ngột.

Lựa chọn độ đo để so sánh RMSE, MAE:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Kịch bản:

Dữ liệu thử nghiệm là phần trăm CPU tiêu thụ đo bằng cadvisor/influxdb, sử dụng jmeter bắn request với chu kỳ 8 tiếng như hình (phần 2).Thử nghiệm trong 24 tiếng, cập nhật lại mạng neural mỗi 8 tiếng 1 lần, tổng cộng 3 lần.

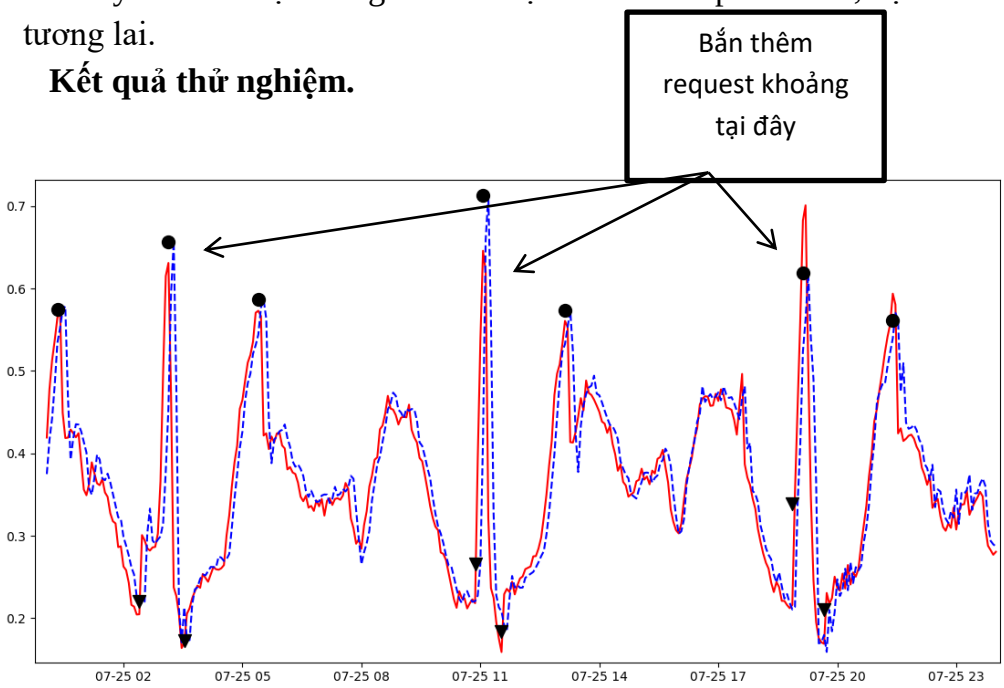
Tại mỗi 8 tiếng, sử dụng jmeter bắn thêm request để giả lập sự thay đổi đột ngột của số lượng request.

Thử nghiệm 2 mô hình riêng biệt:

- 4-4: lấy mẫu dữ liệu trung bình và dự đoán mỗi 4 phút 1 lần, dự đoán 4 phút tương lai.

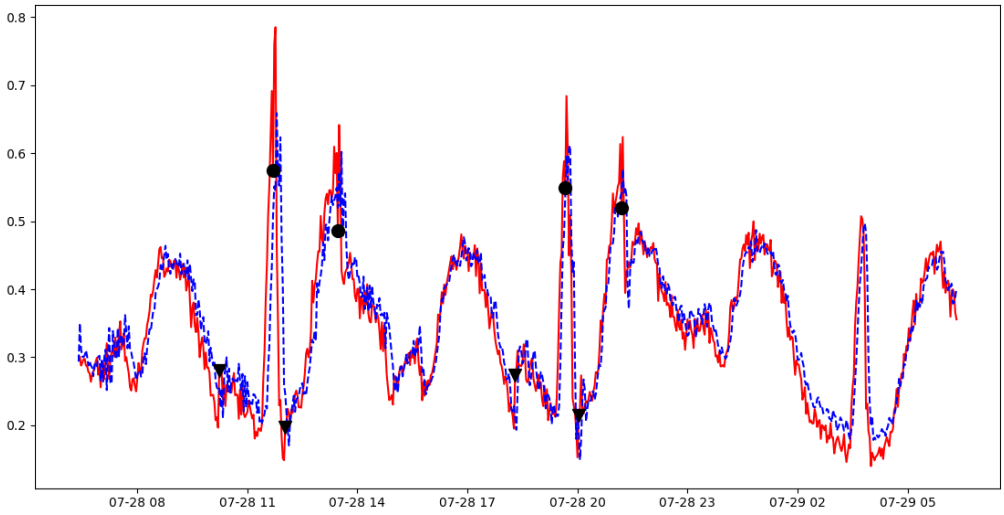
- 2-4: lấy mẫu dữ liệu trung bình và dự đoán mỗi 2 phút 1 lần, dự đoán 4 phút tương lai.

Kết quả thử nghiệm.



Bảng kết quả thử nghiệm mô hình tính trung bình và dự đoán mỗi 4 phút, dự đoán 4 phút tương lai:
Đường đỏ: dữ liệu thật, Đường xanh nét đứt: dữ liệu dự đoán,
Hình tròn đen: thời điểm tạo thêm máy ảo, Hình tam giác đen: thời điểm xóa bớt máy ảo.

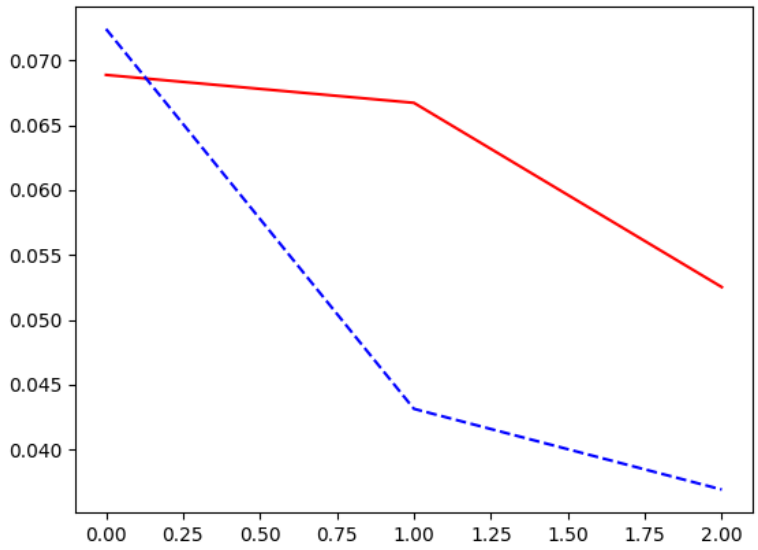
Thông số: periods không tìm được.



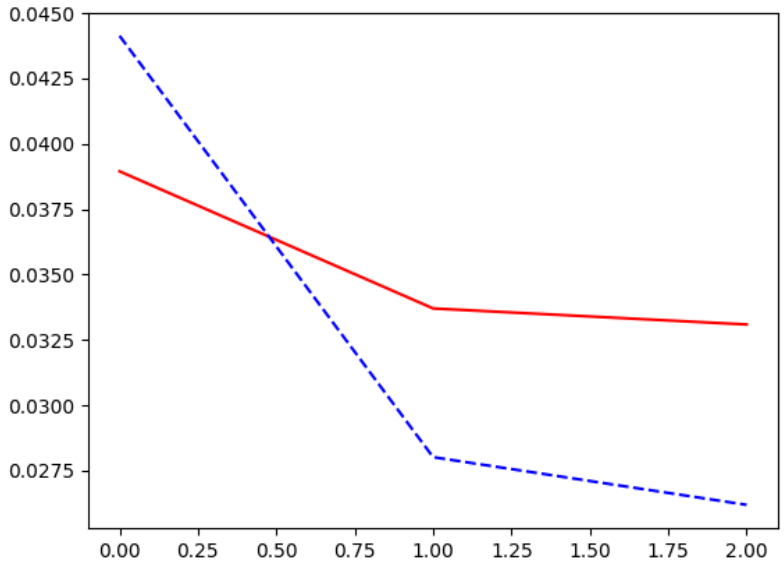
Bảng kết quả thử nghiệm mô hình tính trung bình và dự đoán mỗi 2 phút, dự đoán 4 phút tương lai:
Đường đỏ: dữ liệu thật, Đường xanh nét đứt: dữ liệu dự đoán,
Hình tròn đen: thời điểm tạo thêm máy ảo, Hình tam giác đen: thời điểm xóa bớt máy ảo.

Thông số: periods không tìm được.

So sánh độ chính xác của 2 mô hình.



RMSE sau 3 lần cập nhật mạng neural. Đường màu đỏ: Mô hình 4-4. Đường màu xanh: Mô hình 2-4.



MAE sau 3 lần cập nhật mạng neural. Đường màu đỏ: Mô hình 4-4. Đường màu xanh: Mô hình 2-4.

Nhận xét:

- Chứng minh được dự đoán tốt hơn sau nhiều lần cập nhật (các sai số giảm dần).
- Mô hình 2-4 sau 3 lần cập nhật có tốc độ giảm sai số nhanh hơn là mô hình 4-4, thích hợp hơn trong dự đoán.
- Vấn đề:
    - Khi bắn thêm request (đỉnh thứ 2), dự đoán không bám sát sườn đi lên. Mục đích chính của việc dự đoán ở đây là dự đoán được thời điểm trước khi quá tải để scale (ở đây giả định là 55%). Trong trường hợp không tìm thấy chu kỳ, chỉ dùng 4 điểm sliding window để dự đoán điểm tiếp theo nên dự đoán có độ chính xác chưa cao, và sườn đi lên của dự đoán chậm hơn sườn đi lên của dữ liệu thực nên việc giãn thêm tài nguyên vẫn chậm hơn thời điểm máy ảo bị quá tải.
    - Lần cập nhật 2 dự đoán sườn đi lên bám sát hơn cập nhật 1, cập nhật 3 tốt hơn nữa nhưng nhìn chung giá trị dự đoán ít hơn giá trị thực nên kết quả giãn tài nguyên vẫn chưa tốt.
- Ở mô hình 2-4, các thời điểm tạo thêm tài nguyên tại giá trị (điểm hình tròn đen) nằm trong khoảng 50% đến 58%, khá gần với mong muốn là 55%. Mô hình 2-4 chứng minh nó dự đoán tốt hơn với sự thay đổi dữ liệu tài nguyên tiêu thụ đột ngột (do lượng request tăng đột biến) tốt hơn so với mô hình 4-4.

5.2. Thử nghiệm với dữ liệu không có đột biến.

Mục đích của thử nghiệm này, kế thừa thử nghiệm trước, sử dụng mô hình 2-4 để chứng minh độ hiệu quả của dự đoán co giãn tài nguyên khi không có đột biến.

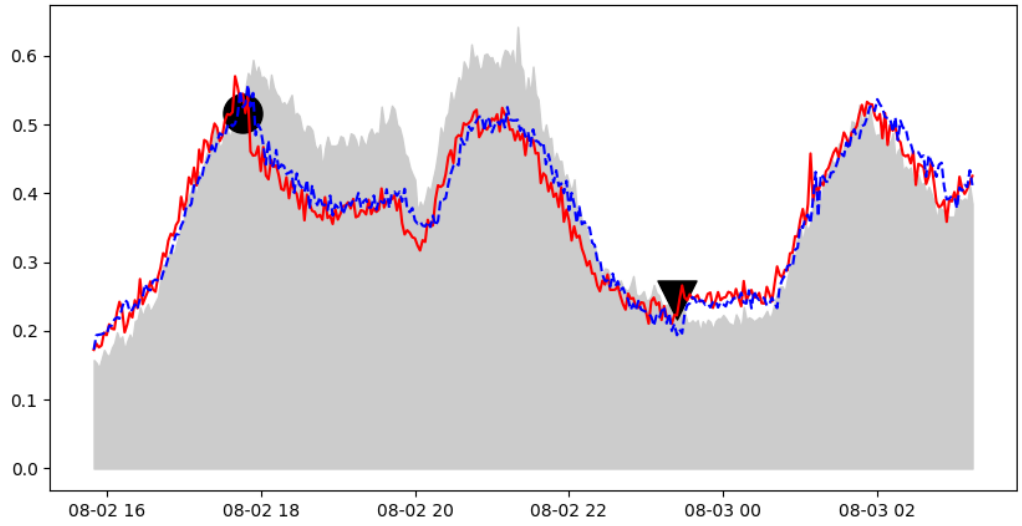
Thiết lập môi trường:

- Môi trường kế thừa từ thử nghiệm trước.
- Tạo một máy ảo đám mây có cấu hình (vCPU, Ram, dung lượng ổ cứng) trên cùng một hạ tầng phần cứng giống như máy ảo thử nghiệm theo dõi, gọi là vm_mirror.
- Sử dụng chương trình duplicator (https://github.com/agnoster/duplicator) để copy thêm request từ jmeter.
  Chạy lệnh:

```
duplicator -f addr_a:port_a -d addr_b:port_b -p port [-i addr]
```

Với addr_a là địa chỉ haproxy của hệ thống, addr_b là địa chỉ máy ảo vm_mirror.

Mỗi 1 request được gửi từ jmeter tới hệ thống đều được copy và gửi tới vm_mirror. Việc này thực hiện với mong muốn có thể tái hiện lại lượng tài nguyên tiêu thụ của máy ảo thử nghiệm nếu như không có dự đoán và giãn tài nguyên tiêu thụ để so sánh.



Hình: Kết quả thử nghiệm:
Đường màu đỏ: dữ liệu tài nguyên tiêu thụ đo được ở máy ảo thử nghiệm.
Đường màu xanh nét đứt: dữ liệu dự đoán.
Nền màu xám: dữ liệu tài nguyên tiêu thụ đo được ở vm_mirror với mong muốn là gần giống với tài nguyên tiêu thụ ở máy ảo thử nghiệm nếu không có mô hình dự đoán.
Hình tròn đen: Thời điểm và giá trị tại thời điểm tạo thêm máy ảo.
Hình tam giác đen: Thời điểm và giá trị tại thời điểm giảm bớt máy ảo.
Độ dài thử nghiệm: 10 tiếng.
Thông số: periods = 0.66 của 24 tiếng, tức là chu kỳ khoảng 16 tiếng.

Nhận xét:

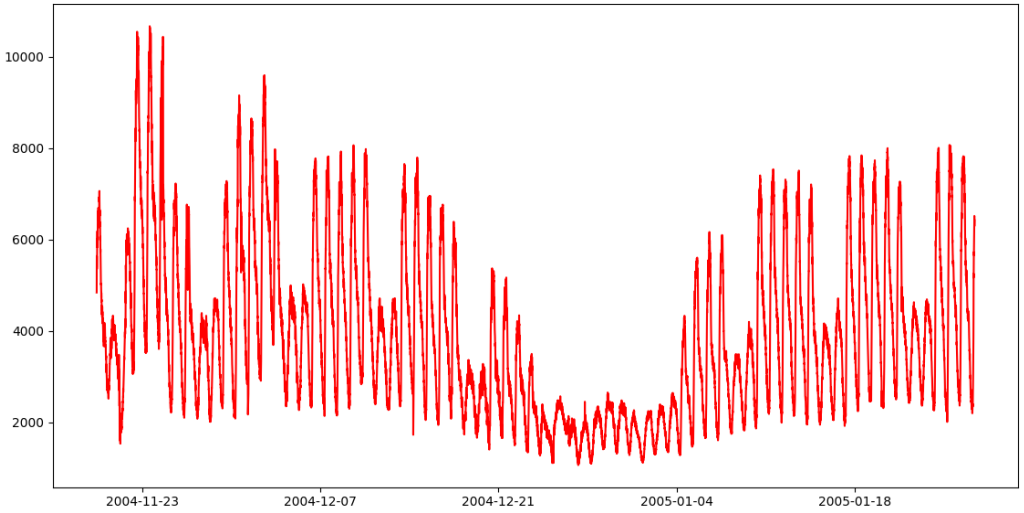
- Kết quả dự đoán trường hợp này đã có sườn đi lên bám sát hơn với dữ liệu thực nên dự đoán khá tốt, máy ảo không bị quá tải (không bị vượt 55% CPU tiêu thụ).

6. Kết luận.

- Phương pháp PD GABP cho độ chính xác tốt trong lý thuyết nhưng việc ứng dụng trong thực tế việc dự đoán quá tải trước để giãn tài nguyên cần phải lưu ý:

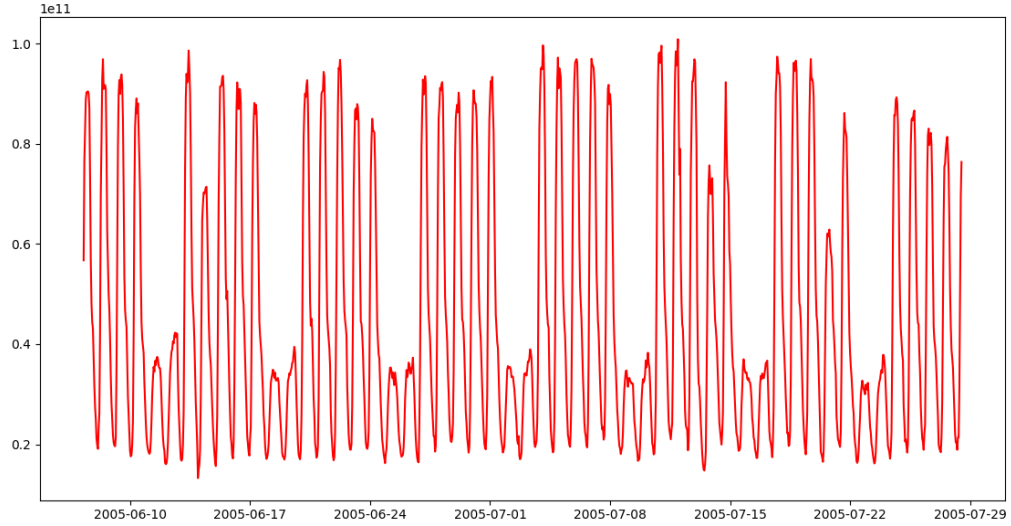
    - Việc dự đoán có độ sai số nhỏ là tốt, nhưng quan trọng hơn là dự đoán sườn đi lên của dữ liệu tài nguyên tiêu thụ phải sát với dữ liệu thật. Điều này mô hình làm khá tốt nếu dữ liệu có phát hiện được chu kỳ, trường hợp khác mô hình vẫn chưa có độ chính xác cao.
- Dự đoán thường xuyên hơn (so sánh giữa dự đoán 2 phút 1 lần và 4 phút 1 lần) sẽ có khả năng phản ứng với lượng request tăng đột biến tốt hơn.
- Điểm hạn chế là mô hình cần phát hiện được chu kỳ thì dự đoán mới chứng minh được là tốt, nếu chỉ sử dụng sliding window để dự đoán thì chỉ ở mức chấp nhận được. Trong thực tế lượng request gửi đến 1 ứng dụng là không thể đoán trước và luôn thay đổi nên khó có thể có được chu kỳ nếu thời gian ngắn. Nhưng về tổng thể lâu dài, dữ liệu vẫn có thể có chu kỳ trong trường hợp lý tưởng. Các ứng dụng trên thực tế có lượng tài nguyên theo chu kỳ là khá phổ biến. Có thể trong thời gian ngắn tính bằng ngày, không thể phát hiện ra chu kỳ, nhưng thời gian dài hơn ví dụ tính bằng tháng hoặc năm, chu kỳ phát hiện được có thể là 1, 2 tháng, hoặc là 1, 2 năm.

Ví dụ về tính chu kỳ trên thực tế:



Hình: Internet traffic data (in bits) from an ISP. Aggregated traffic in the United Kingdom academic network backbone. It was collected between 19 November 2004. Lấy mẫu mỗi 5 phút 1 lần.

Chu kỳ tìm được: Với fs=288, chu kỳ tìm được là 0.989 tức là gần 1 ngày.



Hình: Internet traffic data (in bits) from a private ISP with centres in 11 European cities. The data corresponds to a transatlantic link and was collected from 06:57 hours on 7 June to 11:17 hours on 31 July 2005. Lấy mẫu 1

Chu kỳ tìm được: Với fs=240, chu kỳ tìm được là 2.979 tức là gần 3 ngày.