GIẢM ĐỘ PHỨC TẠP THUẬT TOÁN TÌM PHÁC ĐỒ SỬA LỖI XÓA CỦA MÃ REED-SOLOMON RS(n,k), $n-k=2$

Nguyễn Thị Ngọc Bích[1], Đoàn Thị Thúy Vân[1], Phạm Hữu Khánh[1], Đinh Thị Xinh[1]

Ngày nhận bài: 15/5/2022; Ngày phản biện thông qua: 10/10/2022; Ngày duyệt đăng: 11/10/2022

TÓM TẮT

Các mã Reed-Solomon RS(n,k) là các mã được sử dụng phổ biến nhất hiện nay trong các hệ thống lưu trữ dữ liệu phân tán nhưng chúng có nhược điểm lớn bởi tiêu tốn lượng băng thông khổng lồ khi sửa các lỗi xóa của hệ thống. Phương pháp sửa lỗi xóa cho mã Reed-Solomon bằng cách áp dụng hàm vết do Guruswami và Wootters (Guruswami and Wootters, 2017) đề xuất làm giảm đáng kể băng thông sửa lỗi. Tuy nhiên việc xác định các phác đồ sửa lỗi cho một hoặc nhiều lỗi xóa theo phương pháp này vẫn có độ phức tạp rất lớn ngay cả trên các trường mã hóa nhỏ như F_{16}, F_{64} hay F_{256}. Vì vậy các phương pháp làm giảm độ phức tạp của thuật toán tìm phác đồ sửa lỗi cho mã Reed-Solomon là một vấn đề đáng lưu tâm. Bài báo của chúng tôi đưa ra một số kết quả về giảm độ phức tạp thuật toán tìm phác đồ sửa lỗi xóa cho mã Reed-Solomon, đặc biệt là cho các mã có phần dư $n-k=2$.

***Từ khóa**: Mã Reed-Solomon, hàm vết, lỗi xóa, đặc số của trường.*

1. MỞ ĐẦU

Sửa lỗi xóa (repair erasures) của mã Reed-Solomon trong các hệ thống lưu trữ dữ liệu phân tán (distributed storage systems) với băng thông thấp là vấn đề đang được quan tâm nghiên cứu hiện nay. Các hệ thống lưu trữ dữ liệu phân tán là các hệ thống lưu trữ dữ liệu hiện được sử dụng phổ biến trên thế giới. Mã được sử dụng nhiều nhất trong các hệ thống này là mã Reed-Solomon. Nhiều công ty lớn như Google, Facebook, Microsoft, IBM... đều sử dụng mã Reed-Solomon cho các hệ thống lưu trữ dữ liệu của mình (Dau et al., 2018, Table 1). Lí do cho sự phổ biến này là mã Reed-Solomon giúp tối ưu không gian lưu trữ dữ liệu, nhưng vẫn đảm bảo tính khả dụng (availability) cho dữ liệu được lưu trữ. Trong các hệ thống lưu trữ dữ liệu phân tán, khi một ký hiệu (symbol) lưu trữ trong một server/một nút (node) của hệ thống bị hỏng thì cần khôi phục lại dữ liệu đó bằng cách tải dữ liệu từ các kí hiệu đang được lưu trữ trong các nút không hỏng/nút còn sống (surviving nodes). Lượng dữ liệu cần tải để sửa lỗi được gọi là băng thông sửa lỗi (repair bandwidth).

Phác đồ sửa lỗi mặc định (default repair schemes), hiện được sử dụng để sửa lỗi xóa trong các hệ thống lưu trữ dữ liệu phân tán thực tiễn, tiêu tốn lượng băng thông khổng lồ khi sửa các lỗi xóa. Trong phác đồ sửa lỗi mặc định, hệ thống luôn tải dữ liệu để khôi phục dữ liệu gốc, rồi từ đó khôi phục nút bị hỏng. Cụ thể, để sửa một kí hiệu bị xóa cho mã Reed-Solomon RS(n,k), hệ thống phải tải k kí hiệu. Do đó, nếu mã Reed-Solomon được định nghĩa trên trường F_{2^ℓ} thì dữ liệu cần tải là $k\ell$ bits. Trong một thời gian dài, đây được xem như phác đồ sửa lỗi tối ưu (tải lượng dữ liệu ít nhất từ các nút còn sống) cho các mã Reed-Solomon. Hiện nay, phác đồ này vẫn đang được sử dụng trong các hệ thống lưu trữ thực tế. Chẳng hạn, hệ thống lưu trữ của Facebook f4 sử dụng mã RS$(14,10)$. Trong hệ thống này, khi xảy ra lỗi xóa, chẳng hạn mất 10MB dữ liệu, để sửa lỗi, hệ thống phải tải 100MB (gấp 10 lần dữ liệu bị mất) (Sathiamoorthy et al., 2013). Đặc điểm này của mã Reed-Solomon gây ra chi phí khổng lồ trong việc duy trì tính khả dụng của dữ liệu. Bởi thực tế, lỗi xóa luôn luôn xảy ra trong các hệ thống lưu trữ dữ liệu phân tán, trong đó, hầu hết các trường hợp là một lỗi xóa và hai lỗi xóa (Rashmi et al., 2013).

Một kết quả quan trọng đã được Shanmugam và cộng sự (Shanmugam, K. & et al., 2014) chỉ ra, theo đó $k\ell$ chưa phải là băng thông nhỏ nhất của một phác đồ sửa lỗi cho mã RS(n,k) trên trường F_{2^ℓ}, hay tổng quát hơn là trường F_{q^ℓ}, với q là lũy thừa của một số nguyên tố. Dimakis và cộng sự (Dimakis, A. G. & et al., 2010) chứng minh rằng lượng dữ liệu cần tải để sửa một lỗi xóa thực sự có thể giảm bằng cách xem các kí hiệu như các vector gồm ℓ kí hiệu con (nếu là trường F_{2^8} thì mỗi kí hiệu là một byte gồm 8 bits). Theo đó, hệ thống có thể tải dữ liệu không phải chỉ từ k nút còn sống mà có thể từ nhiều hơn k nút. Tuy nhiên, tại mỗi nút sẽ tải ít hơn ℓ kí hiệu con. Nên tổng dữ liệu cần tải có thể nhỏ hơn $k\ell$. Kết quả trên đã được Guruswami và Wootters (Guruswami and Wootters, 2017) ứng dụng và làm rõ bằng kĩ thuật sửa lỗi sử dụng hàm vết. Đây là một kết quả quan trọng trong nỗ lực giảm băng thông sửa lỗi trong các hệ thống lưu trữ dữ liệu phân tán.

[1]*Khoa KHTN&CN, Trường Đại học Tây Nguyên;*
Tác giả liên hệ: Đinh Thị Xinh; ĐT: 0979085045; Email: dinhthixinh@ttn.edu.vn.