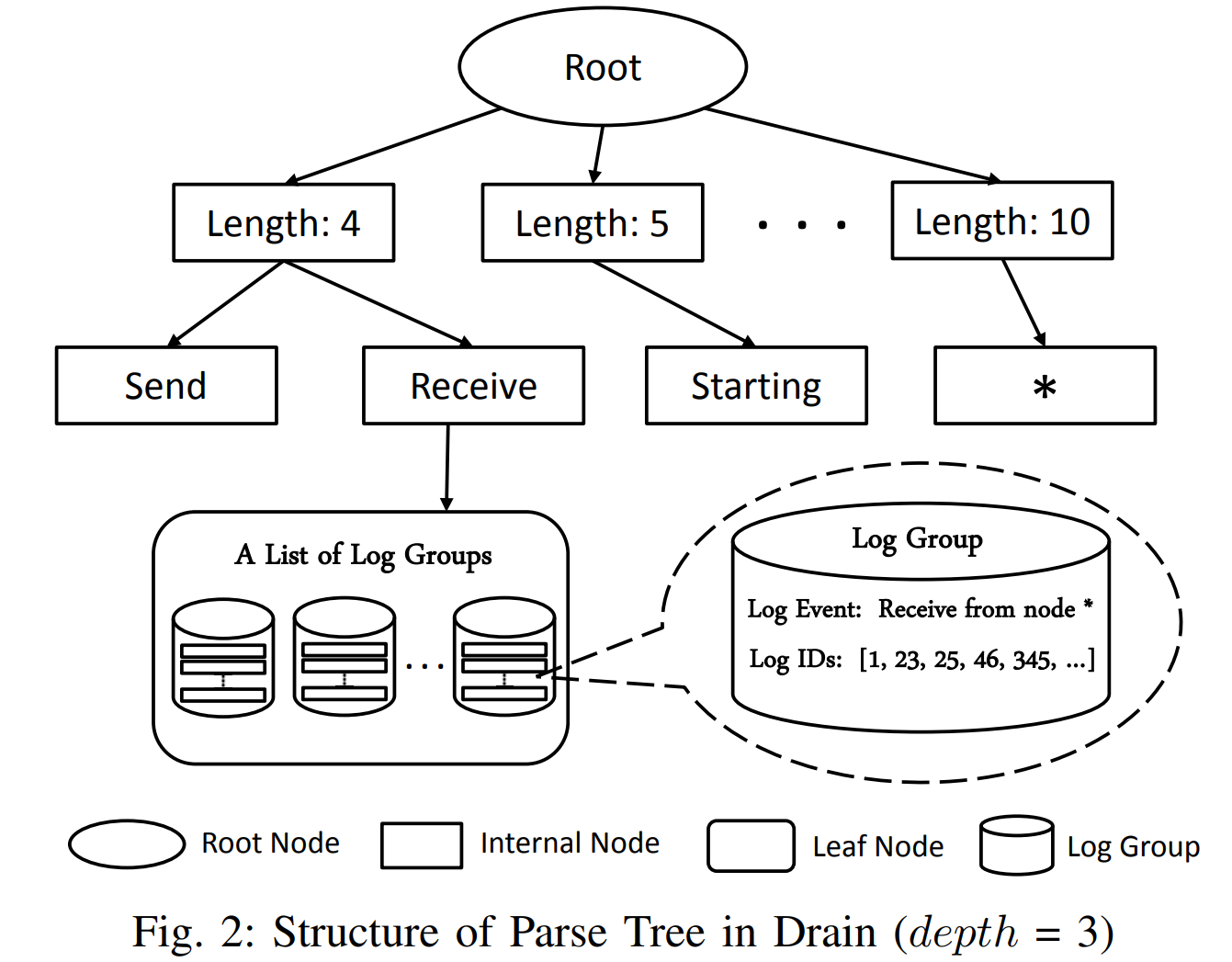
# CẤU TRÚC DRAIN

1.1. Cấu trúc cây tổng thể

Tổng quan: Khi một message log thô đến, Drain sẽ tìm kiếm nhóm log phù hợp nhất cho nó hoặc tạo một nhóm log mới. 🡺 Sử dụng một cây phân tích (parser tree) với chiều sâu cố định để tìm kiếm nhóm log (log group) 🡺 Giúp giới hạn một cách hiệu quả số lượng nhóm log mà các message log cần so sánh



Cây tìm kiếm được minh họa như hình trên, root node (node gốc) nằm ở lớp trên cùng của parser tree, lớp dưới cùng là các leaf node (node lá); các node khác trong cây gọi là internal node. Root node và internal node mã hóa các quy tắc đặc biệt trong quá trình tìm kiếm, chúng không chứa bất kì nhóm log nào. Mỗi đường dẫn trong parser tree đều kết thúc bằng một leaf node, lưu trữ một danh sách các nhóm log.

Mỗi nhóm log có 02 phần: log event và log ID. Log event là mẫu mô tả tốt nhất cho các log message trong nhóm này, bao gồm phần cố định của một message log. Các log ID ghi chép các ID của các message log trong nhóm này. *Chú ý rằng* độ sâu của tất cả các leaf node đều giống nhau và được cố định bởi một tham số depth được định trước. Tham số này giới hạn số lượng node mà Drain truy cập trong quá trình tìm kiếm 🡺 Tăng hiệu suất tìm kiếm. Trong hình trên độ sâu của các leaf node được cố định là 3.

Để tránh việc parser tree bị nổ nhánh (branch explosion) 🡺 Phương pháp sử dụng tham số maxChild, giới hạn số lượng con của một node. Hiểu rõ hơn, một lớp node thứ sẽ có độ sâu là và sẽ có giới hạn số node con của mỗi node là maxChild.

1.1.1. Tiền xử lý dữ liệu thủ công.

Việc tiền xử lý có thể cải thiện độ chính xác của việc phân tích. Cụ thể, Drain cho phép người dùng cung cấp các biểu thức chính quy (regular expression) đơn giản dựa trên nội dung của chúng, chẳng hạn: địa chỉ IP và ID block. Sau đó, Drain loại bỏ các *token* được kết hợp từ message log thô bằng những biểu thức chính quy này (*🡪 Có thể chuyển chúng thành tham số để sử dụng sau này*). *“*Token*” ở đây là các từ, các tham số được truyền vào câu lệnh print khi in ra 01 message log.*

Các biểu thức chính quy được sử dụng trong bước này thường rất đơn giản, vì chúng được sử dụng để phù hợp với các token. Ngoài ra, một tập dữ liệu thường chỉ yêu cầu một số ít biểu thức chính quy (*Trong phương pháp chỉ sử dụng 03 rex*).

1.1.2. Tìm kiếm dựa trên độ dài của message log.

Drain bắt đầu từ root node của parser tree với message log được tiền xử lý. Các node ở lớp thứ nhất (layer node 1) trong parser tree đại diện cho độ dài của các nhóm log (*Số lượng token có trong 01 message log*). Tại bước này, Drain chọn một đường dẫn đến một node trong lớp node thứ nhất dựa trên độ dài của message log được tiền xử lý. *Ví dụ, với message log “Receive from node 4”, Drain sẽ duyệt đến internal log "Length: 4"*. Nghĩa là, các message log với cùng một event log có thể có cùng độ dài message log. Mặc dù có thể có trường hợp các message log với cùng một event log có độ dài khác nhau, điều này có thể được xử lý sau quá trình xử lý.

1.1.3. Tìm kiếm dựa trên các token trước đó.

Tiếp theo, Drain tiếp tục truy cập đến các node của độ sâu tiếp theo để đến leaf node. Bước này dựa trên giả định rằng các token ở các vị trí bắt đầu của một message log có khả năng cao là các hằng số. Cụ thể, Drain chọn internal node tiếp theo dựa trên các token ở các vị trí bắt đầu của message log. (*Ví dụ, với message log “Receive from node 4”, Drain duyệt từ node lớp thứ nhất “Length: 4” đến node lớp thứ hai “Receive” vì đây là token ở vị trí đầu tiên của message log. Sau đó, Drain sẽ duyệt đến leaf node được liên kết với internal node “Receive” và chuyển sang bước 1.1.4*)

Số lượng các internal node mà Drain duyệt trong bước này là , trong đó là tham số giới hạn độ sâu cho tất cả leaf node trong parser tree. Do đó, có lớp được mã hóa với token đầu tiên trong các message log. *Trong hình trên là mô tả chỉ dựa trên token đầu tiên của các message log. Trong thực tế, Drain có thể xem xét nhiều token trước hơn với các thiết lập độ sâu lớn hơn. Lưu ý rằng nếu độ sâu là 2, Drain chỉ xem xét lớp đầu tiên được sử dụng.*

Trong một số trường hợp, một message log có thể bắt đầu bằng một tham số, ví dụ, “120 bytes received”. Những loại message log như vậy có thể dẫn đến sự nổ nhánh của các nhánh trong parser tree vì mỗi tham số sẽ được mã hóa trong một internal node. Để tránh sự nổ nhánh 🡪 Chỉ xem xét các token không chứa chữ số trong bước này. Nếu một token chứa chữ số, nó sẽ phù hợp với một internal node đặc biệt “\*”. Ví dụ, đối với message log trên, Drain sẽ duyệt đến internal node “\*” thay vì “120”. Ngoài ra, tham số được sử dụng để giới hạn số lượng con tối đa của một node. Nếu một node đã có node con, bất kỳ token không phù hợp nào sẽ phù hợp với internal node “\*” trong số tất cả các con của nó.

1.1.4. Tìm kiếm dựa trên độ tương đồng của token

Khi Drain duyệt đên một leaf node, bước 4 được thực hiện. Các leaf node chứa danh sách các nhóm log, các message log trong nhóm log này tuân theo các quy tắc được mã hóa trong các internal node dọc theo các đường dẫn khi Drain thực hiện tìm kiếm. Ví dụ, trong hình trên, log event “Receive from node \*” có độ dài 4, token bắt đầu là “Receive”.

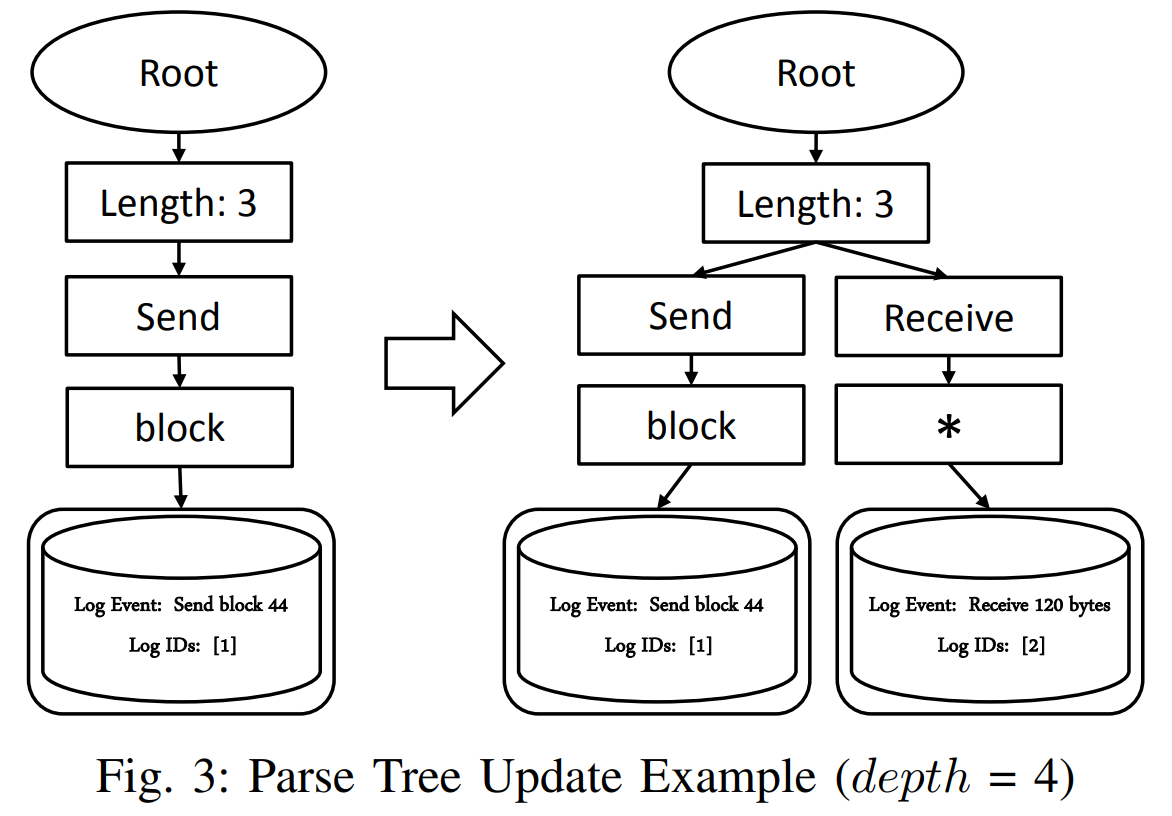
Tại bước này, Drain chọn nhóm log phù hợp nhất từ danh sách nhóm log. Độ tương đồng giữa message log và event log của mỗi nhóm log được tính toán như sau:

Với và đại diện cho message log và event log; là token thứ của chuỗi; là độ dài của message log; hàm được định nghĩa như sau:

Với là hai token. Sau khi tìm thấy nhóm log có giá trị lớn nhất, nó sẽ được so sánh với một ngưỡng độ tương đồng được định trước . Nếu , Drain trả về nhóm làm nhóm log phù hợp nhất. Ngược lại, Drain trả về một flag (Ex, “None” trong Python) để chỉ ra không có nhóm log phù hợp.

1.1.5. Cập nhật cho Parser Tree

Nếu có một nhóm log phù hợp được trả về trong bước 4, Drain sẽ thêm ID log của message log hiện tại vào các ID log trong nhóm log được trả về. Ngoài ra, event log trong nhóm log được trả về sẽ được cập nhật. Cụ thể, Drain quét các token ở vị trí tương ứng của message log và event log. Nếu hai token đó giống nhau, ta sẽ không sửa đổi token ở vị trí đó. Ngược lại, ta cập nhật token ở vị trí đó bằng ký tự đại diện (“\*”) trong event log.



Nếu Drain không thể tìm thấy một nhóm log phù hợp, nó sẽ tạo một nhóm log mới dựa trên message log hiện tại, trong đó các ID log chỉ chứa ID của message log và event log chính xác là message log. Sau đó, Drain sẽ cập nhật parser tree với nhóm log mới. Đơn giản, Drain duyệt từ root node đến một leaf node mà nên chứa nhóm log mới, và thêm các internal node còn thiếu và leaf node tương ứng dọc theo đường dẫn. Ví dụ, giả sử parser tree hiện tại là cây ở phần bên trái của Hình 3, và một message log mới “Receive 120 bytes” đến. Sau đó, Drain sẽ cập nhật parser tree thành cây ở phần bên phải của Hình 3.