Mateusz Małowiecki

Opracowanie noty 2.6 (*Flooding versus random walks*) z rozdziału 2 (*Architectures*) książki *Distributed Systems*[1]

19 marca 2021

[1] Van Steen M., Tanenbaum A.S.: *Distributed Systems 3rd edition.*

Wstęp

Na pierwszy rzut oka, może się wydawać, że algorytm *flooding* jest dużo lepszy niż algorytm *random walks*, gdyż przeszukuje na raz więcej węzłów i jest w stanie szybciej znaleźć rozwiązanie. Jednak w praktyce często mamy do czynienia ze zwielokrotnionymi danymi i badania pokazały, że nawet w przypadku, gdy współczynnik zwielokrotnienia jest niewielki, algorytm *random walks* jest nie tylko efektywny, ale też bardziej wydajny od algorytmu *flooding*.

Obliczenia (algorytm *random walks*)

Załóżmy, że mamy N węzłów, każdy element danych znajduje się na r losowo wybranych maszynach. Poszukiwania polegają na losowym wybieraniu kolejnych węzłów, póki poszukiwany element nie zostanie znaleziony. Jeśli $P[k]$ jest prawdopodobieństwem znalezienia elementu po k próbach, to:

$$P[k] = \frac{r}{N} * (1 - \frac{r}{N})^{k-1} \tag{1}$$

Niech S będzie oczekiwaną liczbą węzłów, które musimy odwiedzić przed znalezieniem żądanej pozycji danych. Wtedy:

$$S = \sum_{k=1}^{n} k * P[k] = \sum_{k=1}^{n} k * \frac{r}{N} * (1 - \frac{r}{N})^{k-1} \tag{2}$$

Stosując proste przekształcenia, można oszacować, że $S \approx N/r$. Możemy zatem zauważyć, że jeśli $r = N$, to $S = 1$, więc algorytm *random walks* jest wtedy lepszy od algorytmu *flooding*. Aczkolwiek gdy r jest dużo mniejsze niż N (przykładowo $r/N = 0.1\%$), to oczekiwana liczba węzłów wyniesie około 1000.

Obliczenia (algorytm *flooding*)

W celu dokonania porównania między algorytmem *random walks* a algorytmem *flooding* załóżmy, że w algorytmie *flooding* pierwszy wierzchołek wysyła komunikat do d wybranych sąsiadów, a każdy następny wierzchołek przesyła go dalej do $d - 1$ wybranych sąsiadów. Wówczas po k krokach osiągniemy co najwyżej

$$R(k) = d * (d - 1)^{k-1} \tag{3}$$

węzłów. Zatem jeżeli wykonamy k kroków (dla takiego k, że $\frac{r}{N} * R(k) \geqslant 1$) to z dużym prawdopodobieństwem znajdziemy węzeł, który zawiera poszukiwany element danych.

Porównanie

Rozważmy jeszcze raz przypadek, kiedy $r/N = 0.1\%$ (czyli $S \approx 1000$). Jeśli w algorytmie *flooding* przyjmiemy $d = 10$ to po czterech krokach osiągniemy 7290 węzłów, czyli zdecydowanie więcej niż 1000. Jednak dla $d = 33$, po zaledwie 2 krokach osiągniemy ok. 1000 węzłów i jednocześnie spełnimy warunek $\frac{r}{N} * R(k) \geqslant 1$. Oczywistą wadą algorytmu *random walks* jest też to, że wdrażanie tego algorytmu może potrwać znacznie dłużej, zanim odpowiedź zostanie zwrócona.