Modelagem de Row Key no HBase para Séries Temporais

A modelagem eficiente de dados no HBase, focado por exemplo para séries temporais, exige uma compreensão profunda de como a arquitetura orientada a colunas gerencia a distribuição de dados. O design da Row Key é o fator determinante para o desempenho, pois o HBase armazena dados lexicograficamente e particiona as tabelas em regiões baseadas nessas chaves.

Um dos principais desafios no armazenamento temporal é o fenômeno de "hotspotting". Isso ocorre quando se utiliza o timestamp (data e hora) diretamente como prefixo da chave. Como o tempo é monotonicamente crescente, todas as novas gravações acabam sendo direcionadas para a mesma região final (a que detém o "tempo atual"). Isso sobrecarrega um único RegionServer, criando um gargalo de escrita e anulando o benefício do processamento distribuído, enquanto os outros servidores do cluster permanecem ociosos.

Para mitigar isso e criar uma chave performática, deve-se buscar uma distribuição aleatória de escritas e, ao mesmo tempo, permitir leituras sequenciais (scan range) eficientes. As boas práticas incluem evitar sequências previsíveis no início da chave.

Existem estratégias consolidadas para a composição da

chave. O uso de "Salting" adiciona um prefixo aleatório à chave, o que distribui bem a carga, mas dificulta a leitura, pois exige a recombinação de dados de várias regiões. O "Hashing" de um atributo fixo (como o ID do sensor) é mais eficaz, pois garante que os dados de um mesmo dispositivo fiquem agrupados. Outra técnica vital para séries temporais é o "Reverse Timestamp" (calculado como Long.MAX_VALUE menos o timestamp atual). Isso faz com que os registros mais recentes apareçam no topo da tabela, acelerando consultas que buscam os dados mais novos.
Considerando o cenário proposto de 10 milhões de sensores com gravações por segundo e a necessidade de consultas por sensor e intervalo de tempo, a modelagem ideal da Row Key seria:
Hash(SensorID) + SensorID + (Long.MAX_VALUE - Timestamp)
Essa escolha se justifica por três motivos. Primeiro, o Hash do SensorID como prefixo garante que os 10 milhões de sensores sejam distribuídos uniformemente entre todos os RegionServers, evitando hotspots, mesmo que os IDs sejam sequenciais. Segundo, ao colocar o SensorID logo após o hash, garantimos que todas as leituras de um único sensor fiquem na mesma região, otimizando a busca. Por fim, o uso do Timestamp

Invertido no final da chave permite que o sistema encontre as leituras mais recentes imediatamente ao iniciar um scan, garantindo baixa latência e alta performance nas consultas solicitadas.