

NSEC/NSEC3のType Bit Mapsについて

DNS BoF 2019

2019/11/28

Toshifumi Sakaguchi

自己紹介

- 坂口 俊文
- Twitter: @siskrn
- GitHub: <https://github.com/sischkg/>
- DNS Summer Day, DNSOPS.JP BoFで発表 <https://dnsops.jp/>

Agenda

1. NSEC/NSEC3のType Bit Mapsの説明
2. PowerDNS RecursorでのType Bit Mapsの実装の問題
3. Type Bit Mapsのテキスト表現
4. Type Bit Mapsとゾーン転送・キャッシュのダンプの話
5. まとめ

Type Bit Maps

NSECレコードは以下の項目を示します。

1. OwnerとNextの間にドメイン名は存在しないこと
2. OwnerにはType Bit Mapsに書かれたタイプのみ存在すること

```
dnsops.jp. 10800 IN NSEC _443._tcp.dnsops.jp. A NS SOA MX AAAA RRSIG NSEC DNSKEY  
CAA
```

- dnsops.jp.と_443._tcp.dnsops.jp.の間にはドメイン名は存在しない
- dnsops.jp.には、A NS SOA MX AAAA RRSIG NSEC DNSKEY CAAレコードが存在する

Type Bit MapsのWire Format(1)

Typeの上位8 bitsをWindow Blockで、下位8bitsをBit Mapで表現

- 0-255のType:
 - Window Block = 0
 - 存在するTypeに対応するBit MapのBitを 1
- 256-511のType:
 - Window Block = 1
 - 存在するTypeに対応するBit MapのBitを 1
- ...

Window Block								Bit Map Length					Bit Map																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511

Type Bit MapsのWire Format(2)

Types = A(1), AAAA(28), CAA(257)の場合

(本来はRRSIGなども含まれるが紙面の都合で省略)

0x00	0x04	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0			
Window Block	bit map length	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

0x01	0x01	0	1	0	0	0	0	0	0
Window Block	bit map length	2	2	2	2	2	2	2	2
		5	5	5	5	6	6	6	6
		6	7	8	9	0	1	2	3

Type Bit MapsのWire Format

Type Bit MapsのWire Formatは、サイズが小さくなるように定義。

PowerDNS Recursorのメモリ使用量の問題

発端

- PowerDNS Recursorを例のファジングにかけると数日後にOOMで強制終了
- PowerDNS Recursorの特徴
 - キャッシュのエントリ数を制限する機能(max-cache-entries)→あり
 - キャッシュのメモリ容量を制限する機能→なし
- メモリ使用量の大きいレコードをキャッシュすると、エントリ数の制限以下でメモリを使い切る

PowerDNS Recursorのメモリ使用量の問題

メモリ使用量の大きいレコードは？

- ヘッダファイル “dnsrecords.hh” からメモリ使用量が大きいclassを探す
- NSECおよびNSEC3

```
class NSECRecordContent : public DNSRecordContent
{
// snip
    DNSName d_next;
    std::set<uint16_t> d_set;
private:
};
```

PowerDNS RecursorにおけるType Bit Mapsの実装

- PowerDNS RecursorはType Bit MapsをC++のstd::set<uint16_t>へ保存
- Type Bit Mapsの1bitとstd::setの1エントリが対応
- つまり全bit(65336bits) を $1 \Leftrightarrow$ std::setに 65536エントリ
- std::setはRed-Black treeを用いて実装(CentOS 7.6のGCC 4.8.5)

メモリ使用量の予測

Red-Black treeのノード(CentOS 7.7 x86_64)は40bytes

```
sizeof( _Rb_tree_node<uint16_t> ) == 40
```

Type(2bytes)のデータを保持するために、40bytes消費

```
struct _Rb_tree_node_base {
    _Rb_tree_color _M_color;           // color           4 bytes
    _Base_ptr _M_parent;               // pointer of parent node 8 bytes
    _Base_ptr _M_left;                 // pointer of left node   8 bytes
    _Base_ptr _M_right;                // pointer of right node  8 bytes
}
// snip
struct _Rb_tree_node : public _Rb_tree_node_base
{
    typedef _Rb_tree_node<_Val>* _Link_type;
    _Val _M_value_field;                // Type                2 bytes
}
```

Type Bit Mapsのサイズ

全てのTypeが存在するときのType Bit Mapsのサイズ・メモリ使用量

DNS Message内のサイズ

= $\langle \text{bit map count} \rangle \times (\langle \text{Window Block} \rangle + \langle \text{Bit Map Length} \rangle + \langle \text{Bit Map} \rangle)$

= $256 \times (1 + 1 + 32)$ bytes

= **8704 bytes**

PowerDNS Recursorでのメモリ使用量

= $\langle \text{node size of red-black tree} \rangle \times 65536 + \langle \text{Overhead} \rangle$ bytes

= $40 \times 65536 + \langle \text{Overhead} \rangle$ bytes

= $2,621,400 + \langle \text{Overhead} \rangle$ bytes

~ **3MB (実測値)**

サンプルコード(https://github.com/sischkg/huge_nsec_response/blob/master/set-uint16_t-x100.cpp)

PowerDNS Recursorのメモリ使用量の問題

キャッシュの制限機能

- BINDやUnboundではキャッシュの量をメモリ使用量で制限
- PowerDNS Recursorではエントリ数で制限

PowerDNS Recursorのキャッシュを細工したNSECでレコードで満たすと、非常に多くのメモリを使用

デフォルトの制限値: `max-cache-entries=1,000,000`

推定メモリ使用量: `3MB x 1,000,000 ~ 3TB`

PowerDNS Recursorのメモリ使用量の問題

キャッシュの制限値を3MB/エントリをもとに設定すると、
十分なキャッシュを保持することができない

キャッシュサーバのメモリ: 512GB

キャッシュ可能なエントリ数: $512\text{GB} / 3\text{MB} = 170,666$

PowerDNS Recursorのメモリ使用量の問題

対策: PowerDNS Recursor 4.2.0にバージョンアップ

- 4.2.0ではType Bit MapsのType数が200に達すると、コンテナを `std::set<uint16_t>` から `std::bitset<65536>` へ変更
- Pull Request: Use a bitset for NSEC(3) records with lots of types
#7345 (<https://github.com/PowerDNS/pdns/pull/7345/commits/27d4a65bf8d908b02ea84e662683d31b077306ab>)
- `std::bitset`を利用することで、メモリ使用量をエントリあたり2KB程度まで減少

PowerDNS Recursorのメモリ使用量の問題のまとめ

- PowerDNS RecursorはType Bit Mapsをキャッシュするときのメモリ使用量が多い
- エントリ数制限値が大きい場合は、メモリを使い切る可能性がある
- 4.2.0で修正

Type Bit Mapsのテキスト表現

テキスト表現

- dig/drillなどの出力
- ゾーンファイル
- キャッシュのダンプ

NSECのテキスト表現の例

```
example.com. IN NSEC  dns01.example.com. A NS SOA MX  
RRSIG NSEC DNSKEY
```

Type Bit Mapsのテキスト表現

Type Bit Mapsの全てのbitを1としたNSECレコード

```
example.com. 3600 IN NSEC a.example.com. RESERVED0 A NS MD MF CNAME SOA MB MG MR NULL WKS PTR  
HINFO MINFO MX TXT RP AFSDDB X25 ISDN RT NSAP NSAP-PTR SIG KEY PX GPOS AAAA LOC NXT EID NIMLOC  
SRV ATMA NAPTR KX CERT A6 DNAME SINK OPT APL DS SSHFP IPSECKEY RRSIG NSEC DNSKEY DHCID NSEC3  
NSEC3PARAM TLSA SMIMEA TYPE54 HIP NINFO RKEY TALINK CDS CDNSKEY OPENPGPKEY CSYNC  
...  
TYPE65530 TYPE65531 TYPE65532 TYPE65533 TYPE65534 TYPE65535
```

https://raw.githubusercontent.com/sischkg/huge_nsec_response/master/nsec_response.txt

1レコード当たり約 640KB

ゾーン転送で受信したゾーン

BINDでは、ゾーン転送で受信したゾーンのファイル形式

- 9.8.0未満: テキスト
- 9.8.0以上で"masterfile-format text;": テキスト
- 9.8.0以上で"masterfile-format"未指定: raw

BIND9の最新動向(https://dnsops.jp/event/20120901/BIND9_update-1.pdf)

1000個のNSECレコードを持つゾーンファイルのサイズ

- テキスト形式: 645 MB
- raw形式: 8.8 MB

キャッシュのダンプ

フルリゾルバでは、以下のようにキャッシュの内容を出力することが可能

BIND:

```
rndc dumpdb
```

Unbound:

```
unbound-control dump_cache > /tmp/dumpdb.txt
```

PowerDNS Recursor:

```
rec_control dump-cache /tmp/dumpdb.txt
```

キャッシュのダンプ

1000レコード当たりのダンプファイルのサイズ

- BIND: 約640MB
- Unbound: 約1.1MB
- PowerDNS Recursor: 約1.3GB

想定したサイズより大きいダンプファイルが生成される可能性

- BINDのAdministrator Reference Manualに記載を依頼
- rndc dumpdb output may be larger than expected - documentation update request(<https://gitlab.isc.org/isc-projects/bind9/issues/795>)

キャッシュのダンプ(Unbound)

Unboundでは、Type Bit Mapsをすべて出力しないため、ダンプファイルが小さい

```
1.example.com. 3215 IN NSEC a.1.example.com. A NS MD  
MF CNAME SOA MB .... TYPE152 TYPE1;rrset 10415 1 1 7 0
```

キャッシュのダンプ(PowerDNS Recursor)

PowerDNS Recosor:

- キャッシュをダンプ中に応答しない時がある
- Threadごとにキャッシュを持っているらしく、同じレコードが複数回出力されるため、ダンプファイルが大きい

rec_control(https://doc.powerdns.com/recursor/manpages/rec_control.1.html)

Dumps the entire cache to *FILENAME*. This file should not exist already, PowerDNS will refuse to overwrite it. **While dumping, the recursor will not answer questions.**

Typical PowerDNS Recursors **run multiple threads, therefore you'll see duplicate**, different entries for the same domains. The negative cache is also dumped to the same file. The per-thread positive and negative cache dumps are separated with an appropriate comment.

Type Bit Mapsのテキスト表現のまとめ

- Type Bit Mapsのテキスト表現は、非常に大きなサイズになる場合がある
- ゾーン転送後のゾーンファイルや、キャッシュのダンプファイルのサイズも大きくなる
- ダンプファイルやゾーンファイルの保存先は、ファイルシステムに十分な余裕が必要

まとめ

- PowerDNS Recursorは特殊なNSEC/NSEC3をキャッシュするときのメモリ使用量が多い
- エントリ数制限値が大きい場合は、メモリを使い切る可能性がある
- Type Bit Mapsのテキスト表現は、非常に大きなサイズになる場合がある
- そのためゾーン転送後のゾーンファイルや、キャッシュのダンプファイルのサイズも大きくなる
- ただし、特殊なNSEC/NSEC3を作成して実際に実行する人が存在するかは不明