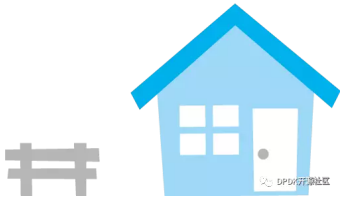
# Hyperscan与Snort的集成方案

原创 DPDK开源社区 2017-04-09

作者 王翔





## 概况

Hyperscan作为一款高性能的正则表达式匹配库，非常适用于部署在诸如DPI/IPS/IDS/NGFW等网络解决方案中。Snort (https://www.snort.org) 是目前应用最为广泛的开源IDS/IPS产品之一，其核心部分涉及到大量纯字符串及正则表达式的匹配工作。本文将重点介绍如何将Hyperscan集成到Snort中来显著提升Snort的总体性能。具体集成代码已公开在https://01.org/hyperscan。

## Snort简介

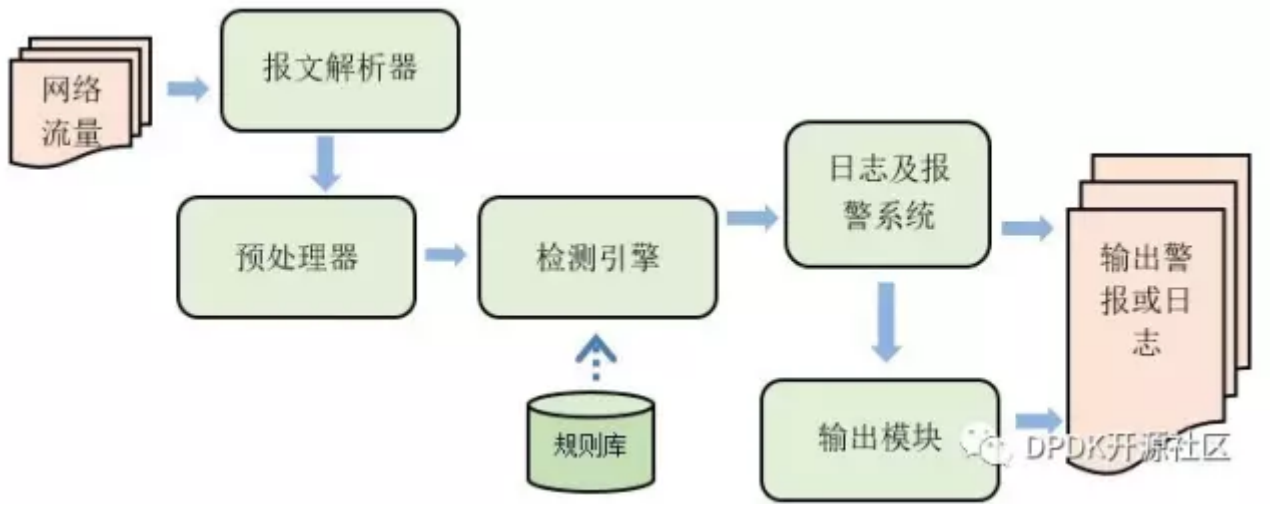



图1：Snort 架构图

如图1所示，Snort主要分成五个部分。报文解析器负责从不同的网络接口接收到报文，并对报文内容进行初步的解析。预处理器是对解析过的报文进一步处理的插件，其功能包括HTTP URI归一化，报文整合，TCP流重组等。检测引擎是Snort当中最为核心的部分。它根据现有的规则，对报文数据进行匹配。匹配的性能对Snort总体性能起着至关重要的作用。假如匹配成功，则依据规则中定义的行为通知日

志及报警系统。该系统可输出相应的警报或者日志。用户也可以定义输出模块来以特定形式（例如数据库，XML文件）保存警报或日志。

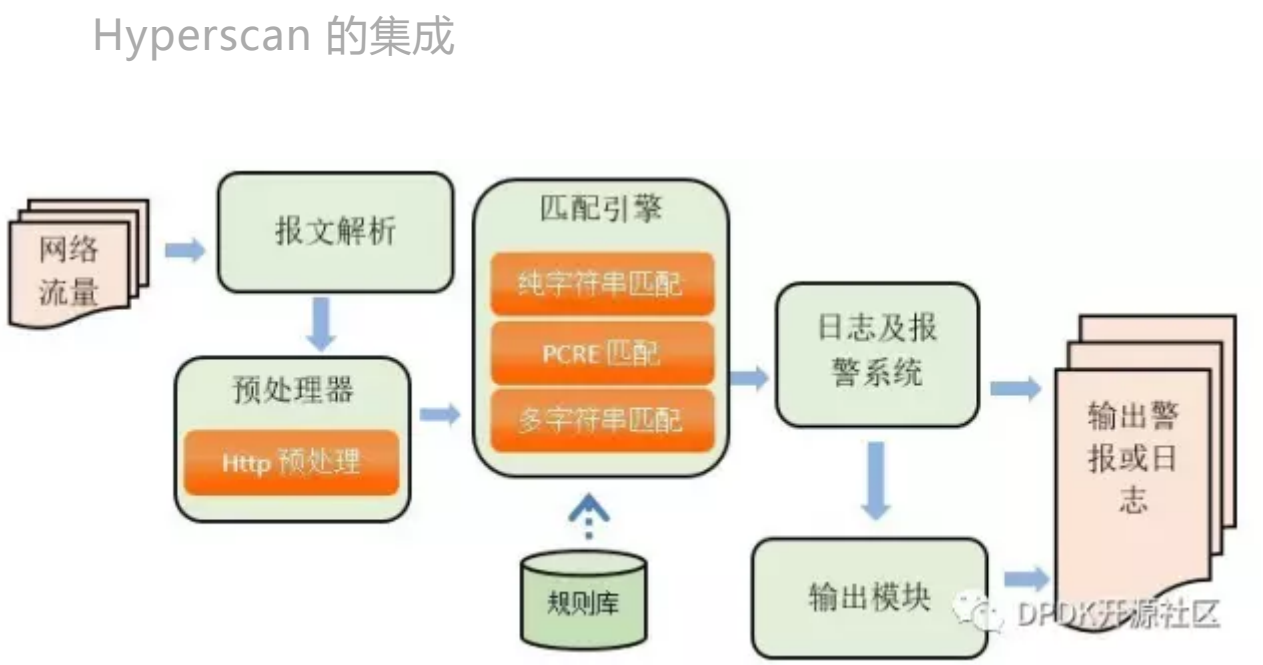



图2：Hyperscan与Snort 的集成

如图2所示，Hyperscan与Snort的集成主要集中在以下四个方面：

### 纯字符串匹配

用户可以在Snort规则中定义匹配特定的字符串，并在相应报文中寻找该字符串。Snort中采用了Boyer-Moore算法进行匹配。我们用Hyperscan对这一算法进行替换以提升匹配性能。

### PCRE匹配

Snort中使用了PCRE来作为正则表达式匹配的引擎。Hyperscan兼容了PCRE的语法规则，但不支持少数回溯及断言语法。但是Hyperscan本身自带有PCRE的预处理功能(PCRE Prefiltering)，可以通过对PCRE规则进行变换以兼容Hyperscan。实际规则产生的匹配是变换后的规则所产生匹配的子集。因此可以使用Hyperscan进行预先扫描，若不产生匹配则实际规则也无匹配。若产生了匹配，可以通过PCRE的扫描来确认是否有真正的匹配。由于Hyperscan的总体性能高于PCRE，Hyperscan的预先过滤可以避免PCRE匹配带来的过大时间开销。

### 多字符串匹配

Snort中另外一个重要的匹配过程是多字符串的匹配。多字符串的匹配可以快速过滤掉无法匹配的规则以减少需要逐条匹配的规则数从而提升匹配的性能。Snort中使用了Aho-Corasick算法进行多字符串的匹配。我们用Hyperscan替代了这一算法并且带来了显著的性能提升。

### Http预处理

除了引擎的匹配算法的集成，我们在预处理器中也添加了Hyperscan。在做Http预处理时，我们利用了Hyperscan搜索相关关键字来进一步加速预处理的流程。

## 性能数据

我们选取了Snort自带的VRT 规则（8683条）作为测试规则，同时以存有真实网络流量信息的PCAP文件作为输入进行测试。图3展示了在Broadwell-EP平台下，原生Snort和经过Hyperscan加速的Snort在单核单线程下的性能对比。我们可以看到，Hyperscan极大提升了Snort的匹配性能，总体性能约是原始Snort性能的6倍。

另外，我们对原生Snort与经过Hyperscan优化后的Snort在内存消耗方面进行了比较。由于原生Snort依赖于Aho-Corasick算法，需要将所有规则转化成Trie树结构，因此占用较大的内存。而Hyperscan拥有自身优化过的匹配引擎进行匹配，大量减少了匹配过程中对内存的消耗。如图4所示，在这个测试中，总体上原始Snort所占用的内存是经过Hyperscan优化后的Snort的12倍。

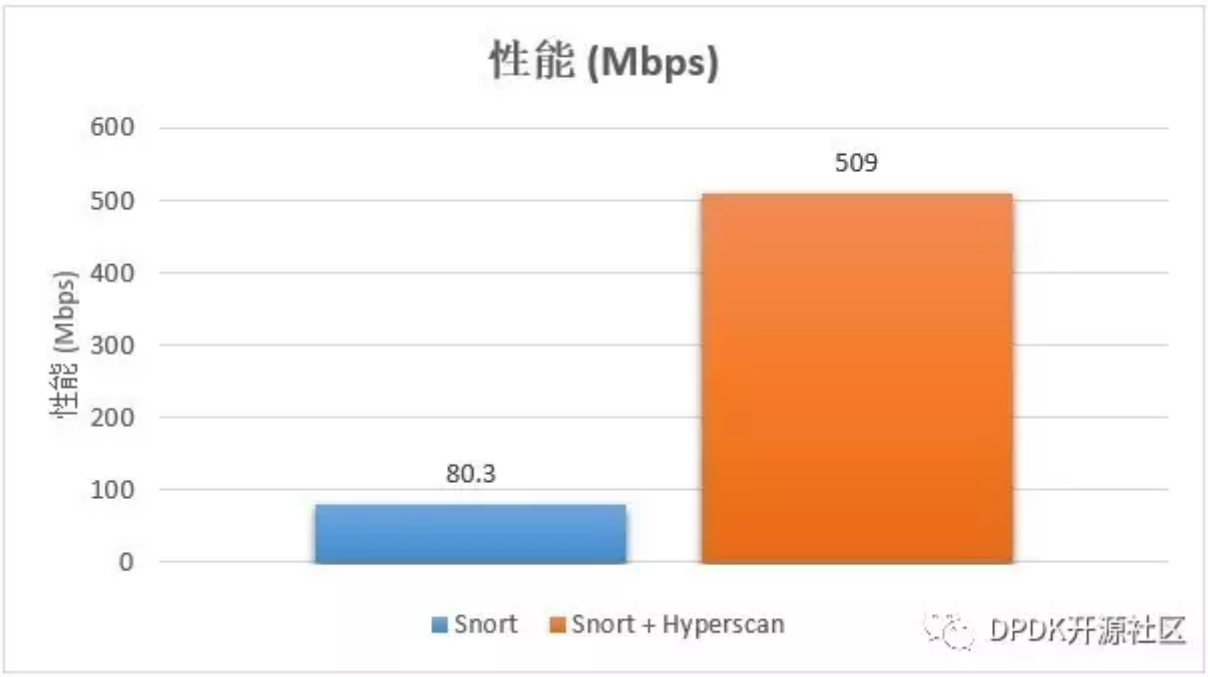



图3：原始Snort与集成了Hyperscan的Snort性能对比

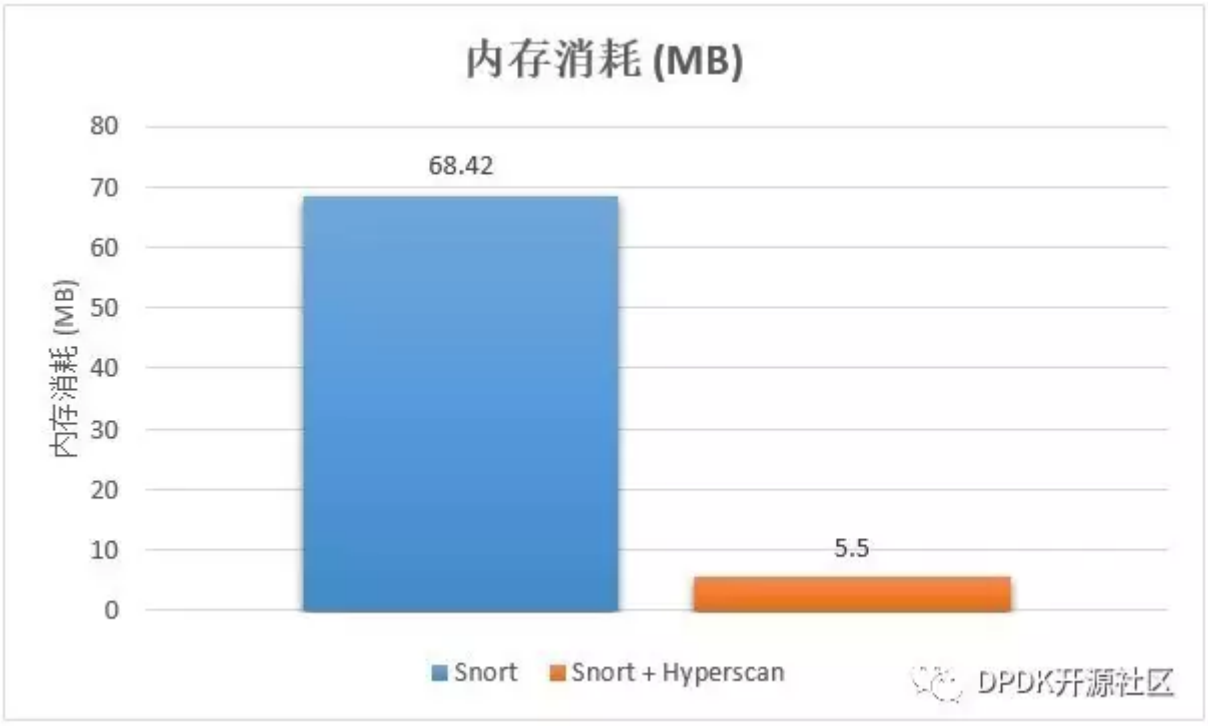



图4：原始Snort与集成了Hyperscan的Snort内存消耗对比

## 总结

经过Hyperscan集成后的Snort不管在总体性能还是内存消耗上都远远优于原始Snort。由此，Hyperscan展现了大规模规则匹配的强大能力，非常适用于以规则匹配为核心的DPI/IDS/IPS/NGFW等产品中。

## 作者简介

王翔，英特尔软件工程师，负责Hyperscan研发。主要研究领域包括正则表达式匹配，深度报文检测等。


**DPDK开源社区 精选文章**

DPDK中的memcpy性能优化及思考

DPDK开源社区加入Linux基金会大家庭了

Hyperscan的模式选择

欢迎搭乘Hyperscan号极速列车~

Hyperscan Release 4.4.0

DPDK Release 17.02

无锁队列详细分解 — 顶层设计

从计算机架构师的角度看DPDK性能

基于virtio-user的新exception path方案

玩物志 | 什么！DPDK在盒子里？

关于DPDK Cryptodev，你不得不明白的几点！