sPIN: High-performance streaming Processing in the Network（阅读笔记）

刘涛
中科院计算所

版本:0.1

更新:June 20, 2019

摘要

要点：

- 优化通信对大规模计算很重要。
- 目前网卡上的处理器的数据传输能力很强，但是功能单一（比如 RDMA）。
- sPIN：一个编程模型，a portable packet-processing network acceleration model（类似于使用 CUDA 或 OpenCL 加速计算）。将 simple packet processing offload 到网卡上。
- 实现的功能：
  - offloaded messsage matching
  - datatype processing
  - collective communications

1 Motivation

用 slow but energy-efficient processors 构建 highly-scalable computing system 给 interconnection netowrk 带来更大的挑战。

最近在 bandwidth 和 latency 方面的工作倾向：remove CPU from packet processing path。

specialized data processors in modern NIC（Network Interface Cards）：RDMA；几十 GB/s，sub-microsecond 延迟。

RDMA 只在不同网络端点上的进程的内存空间之间传输数据。

如果 Tb/s 的网络到来，会有瓶颈：现在的 CPU 访问 L3 Cache 需要 10 15ns，但是 400Gb/s 的 NIC 每 1.2ns 就能传输 64B（一个 cache line）。

最主要的问题是：packets 只是简单的存放在 main memory，然后许多应用程序再对消息进行 analysis 和 rearrange。这个过程造成一个 barrier，类似于 pre-RDMA packets processing：但是 CPU 是用来计算的，并不擅长 message processing。而且这些数据会 pollute caches，将真正用来计算的数据“挤出”cache。由于没有良好的接口，NIC 上的 highly-optimized data-movement cores 只能像瞎子一样把数据扔进 host memory。

本文提出的解决方案：sPIN（streaming Processing in the Network），将 RDMA 和 receiver-based matching 扩展到由 data-movement 主导的简单处理任务。设计了统一接口，可以 specify kernels，在 NIC 上执行。

kernel offload data-movement-heavy tasks。tasks 处理 incoming messages，只需要有限的 local state。

concepts：packetization，buffering，packet steering。

开发者定义 handlers：

- header：header handler code
- packet content handler code
- post-message handler code

在 NIC 的 HPU（handler processing units）上对匹配的 packets 进行处理。

handler code 在连接建立阶段传递，这样单进程就可以对不同的连接指定不同的 handler。

本文主要内容：

- design of an acceleration system for NIC offload
- outline a microarchitecture for offload-enabled smart NICs
- design a cycle-accurate validated simulation environment（整合 network，offload-enabled NIC 和 CPU）
- 分析实例：并行应用、分布式数据管理系统
- speedups

1.1 Background

相关技术：

1. active messages（AM）：
   - AM 针对 full messages；sPIN 则是基于 per-packet 的流式处理。
   - AM 将消息缓存在 host memory；sPIN 主要将消息放在 fast buffer memory on NIC，尽量减少对 host memory 的访问。
   - 在 AM 上一个 message 是 atomic 的（message 到达后，handler 对整个 message 处理）；sPIN 则是每次对消息的一部分（packets）进行原子的操作。

sPIN 比 AM 更靠近 packet processing system。

硬件支持：Intel IXP、Netronome NFP。

编程模型支持: SDN（software defined networking）、P4、FlexNIC。

2 Processing in the Network

sPIN 的中心思想基于：网络设备将消息切分为 packets 进行传输。

开发者定义的 handler functions 处理一组 packets（逻辑上构成一个 message），handlers 被调度到 HPUs 上执行。

header packet：标识 message，steering into memory。

sPIN 通过 NIC-based runtime system 来判断哪些 packets 属于同一个 message，之后对这些 packet 用三种定义好的 handlers 处理。

注 *packet 的到来和处理都是乱序的。*

HPU memory 的管理通过：host OS 和 user-level application。

handler 的调用不需要通过 host OS。handler 一般只包含几百条指令，可以直接存放在 NIC 上的 fast memeory。

handlers 可以通过 shared memory 通信，但是它们不能管理内存，而且它们的 local memory 直接通过物理地址寻址。

handlers 用 C/C++ 编写，只能编写简单的 code，不能使用系统调用和复杂的库。

handler 没有 time-exceeded 机制，programmer 需要保证 handler 的操作是 line-rate，否则可能造成 NIC stall 或丢包。启动 handler 的 cost 要很低，假设一个 packet 到达 one cycle 之内就能启动。向 handler 传参的 overhead 也要很低。

handler 可以通过 DMA 访问 host memory，但是这样做的代价比较大。handler

是 non-blocking 的，如果 handlers 在等待 host memory，可以将它们调度出 HPUs 切换到其他的 handlers。

handlers 可以生成两种 messages：

1. messages originating from HPU memory：只包含一个 packet，可以 block HPU thread 直到 delivered
2. messages originating from host memory：放进 normal send queue，nonblocking。

3 A COMPLETE sPIN INTERFACE

As an example to demonstrate all sPIN features, we use the Portals 4 network interface because it offers advanced receiver-side steering (matching), OS bypass, protection, and NIC resource management.

3.1 Overview of Portals 4

选择 logically addressed 和 matched mode。

sPIN 对 Portal 4 进行整合和扩展提供更强大的 message steering、protocol implementation、packet processing。

3.2 A sPIN Interface for Portals 4

derive a candidate Portals 4 (P4) interface called P4sPIN.

每个 packet handler 都关联一个特定的 ME（mathcing entry），扩展了 PtlMEAppend 调用。

通过 PtlHPUAllocMem 分配 HPU memory，如果多个 incoming messages 匹配的 ME 对应同一段 HPU memeory，handlers 需要进行并发控制。

3.2.1 Header Handler

called once per message。

header handler 完成之前不会启动其他的 handlers。

User-defined headers are declared statically in a struct to enable fast parsing in hardware. pre-defined headers could be in special registers while user-defined headers can reside in HPU memory。

3.2.2 Payload Handler

在 header handler 之后调用，for packets carrying payload。

payload 不包括 user-defined header。

不同的 payload handler 可以并行执行，由 programmer 负责并发控制。

PTL_NUM_HPUS 限制可以同时执行的 handlers 的个数。

PTL_MY_HPU 可以指定 handler 在哪个 HPU 上执行。

handler 开始执行后就不能 migrate 到其他的 HPU 上。

The flag flow_control_triggered indicates that flow control was triggered during the processing of this message and thus some packets may have been dropped without being processed by payload handlers.

3.2.3 Completion Handler

called once per message after 所有 header handlers 和 payload handlers 完成，before 向 host 发送 completion event。

used for final data collection 或 cleanup tasks。

4 PROTOTYPING sPIN

NISA（Network Instruction Set Architecture）

两种作为 NISA 的 sPIN 的原型：

1. discrete network card（dis）：跟 CPU 进行 chip-to-chip 互联（比如 PCIe）。
2. integrated network card（int）：跟 CPU cores 在同一个 chip 上，通过 fast singnaling protocol（比如 AXI）进行互联。

4.1 HPU Design

assume NIC architecture can be re-used，sPIN 可以理解为为每个应用在 NIC 上安装客制化的固件。

HPU 访问 local memory 和 packet buffer 要尽量快，HPU memory is not cached。

大多数 HPU 指令单周期完成，handler 在 packet 到达或之前 handler 完成后立即调用。handler is pre-loaded and pre-initialized. 可以通过多线程高效利用 HPU 执行单元，当 handler thread 等待 DMA 时可以 deschedule，调度其他 threads。

handler 的执行会延迟正在处理的 messages –> 需要足够的 HPU core 在全带宽状态下工作，而且需要额外的内存空间。

可以根据 Little’s law 计算 memory overhead 的大小。通过增加空间掩盖延迟，可以达到几微秒。

4.2 Simulation Setup

两个开源的 simulators：

1. LogGOPSim：模拟并行分布式内存应用的网络。可以模拟 MPI 应用，已经在 IB 集群上调校和验证过。
2. gem5：模拟各种 CPU 和 HPU 配置。cycle-accurate。

4.3 DMA and Memory Contention

4.4 Microbenchmarks

5 USE CASES FOR sPIN

5.1 Asynchronous Message Matching

5.2 MPI Datatype

5.3 Distributed RAID Storage

5.4 Other Use Cases

6 RELATEDWORK

7 DISCUSSION

8 SUMMARY AND CONCLUSIONS

9 模板设计

此模板设计的初衷是为了记录笔记，在 2013 年开始构想，初版我们设计了非常美观的定理环境，并设计了 3 套不同的颜色主题。但我们发现在实际记笔记的时候，过多的定理区块使得整个文章并不是非常美观，所以我们把 ElegantNote 更新为 ElegantBook 模板，在后面被用户熟知。而 ElegantNote 的设计自此停止。

2018 年，在被一些用户“催更”之后，ElegantBook 迎来重大更新，原先浮动的定理环境用 tcolorbox 全部改写。时至今日，ElegantBook 版本为 3.05。之后，我们便想把 ElegantNote 也彻底更新下，放弃 ElegantBook 中的定理环境设计，改用更为紧凑，更加朴素的定理环境，设计更适合笔记记录和笔记阅读的 LaTeX 模板。

在一些朋友的建议和启发下，我们基于标准的 LaTeX 文类 article 重新设计了新版 ElegantNote 模板，在此特别感谢！新模板有下面几个特性：

- 添加护眼模式，颜色为绿豆沙颜色，默认为白色背景；
- 适配不同设备，包括 Pad（默认），Kindle，PC（双页），通用（A4）；
- 5 套颜色主题，分别是：green（默认），cyan，blue，sakura，black；
- 语言模式支持：中文（默认），英文；
- 支持 PDFLaTeX 和 XeLaTeX 编译；
- 更加美观的图表标题格式，列表环境，数学字体等。

9.1 护眼模式

本模板增加了护眼模式，默认为不开启，开启的方法如下：

```
\documentclass[geye]{elegantnote} % or
\documentclass[mode=geye]{elegantnote}
```

评论 此次更新只添加了护眼模式(绿豆沙色背景),如果您有希望增加其他颜色,可以在 *Github/ElegantNote* 创建 *issues* 或者提交 *pull request*。

9.2 设备选择

为了让笔记方便在不同设备上阅读,免去切边,缩放等操作,本模板适配不同的屏幕大小,分别为 Pad,Kindle,PC,A4。不同屏幕的选择为

```
\documentclass[device=pad]{elegantnote}
\documentclass[device=kindle]{elegantnote}
\documentclass[device=pc]{elegantnote}
\documentclass[device=normal]{elegantnote}
```

注 也可以采取直接赋值的方法选择屏幕,比如:

```
\documentclass[pad]{elegantnote}
\documentclass[kindle]{elegantnote}
\documentclass[pc]{elegantnote}
\documentclass[normal]{elegantnote}
```

注 如果想要得到普通的 *A4* 纸张大小的 *PDF*,需要选择 *device=normal*,而不是选择 *device=pc*,因为 *device=pc* 实际上设置的是电脑双页模式。

9.3 颜色主题

本模板内置 5 套颜色主题,分别是 green,cyan,blue,sakura,black。其中 green 为默认颜色主题,如果用户不需要彩色,可以选择 black 主题。颜色主题的启用方法和之前一样:

```
\documentclass[green]{elegantnote}
```

```
\documentclass[color=green]{elegantnote}
...
\documentclass[black]{elegantnote}
\documentclass[color=black]{elegantnote}
```

9.4 语言模式

本模板内含两套语言环境，改变语言环境会改变图表标题的引导词（图，表），文章结构词（比如目录，参考文献等），以及定理环境中的引导词（比如定理，引理等）。不同语言模式的启用如下：

```
\documentclass[cn]{elegantnote}
\documentclass[lang=cn]{elegantnote}
\documentclass[en]{elegantnote}
\documentclass[lang=en]{elegantnote}
```

注 不管选用中文环境还是英文环境均可输入中文。另外如果在笔记中使用了抄录环境（*lstlisting*），并且其中包括了中文，请务必使用 *XeLaTeX* 编译。本说明文档也只能通过 *XeLaTeX* 编译。

9.5 编译方式

本模板支持两种编译方式，PDFLaTeX 和 XeLaTeX，选用 PDFLaTeX 编译的话，如果用到了中文，则会调用 ctex 宏包，而如果选用 XeLaTeX 编译的话，则会调用 xeCJK 宏包。模板测试环境为 Win10 + TEX Live 2018，设定的字体为 Windows 中的宋体、楷体、黑体、微软雅黑等。如果你的电脑是 Mac/Linux 系统，而且采用 XeLaTeX 编译的话，请把 elegantnote.cls 中字体改为自己系统的字体。

9.6 定理类环境

此模板采用了 amsthm 中的定理样式，使用了 4 类定理样式，所包含的环境分别为

- **定理类**：theorem，lemma，proposition，corollary；
- **定义类**：definition，conjecture，example；
- **备注类**：remark，note，case；
- **证明类**：proof。

评论 *在选用 lang=cn 时，定理类环境的引导词全部会改为中文。*

10 写作示例

我们将通过三个步骤定义可测函数的积分。首先定义非负简单函数的积分。以下设 E 是 $\mathcal{R}^n$ 中的可测集。

定义 10.1 (可积性) *设 $f(x)=\sum_{i=1}^{k} a_i \chi_{A_i}(x)$ 是 E 上的非负简单函数，其中 $\{A_1, A_2, \ldots, A_k\}$ 是 E 上的一个可测分割，$a_1, a_2, \ldots, a_k$ 是非负实数。定义 f 在 E 上的积分为*

$$\int_E f dx = \sum_{i=1}^{k} a_i m(A_i). \tag{1}$$

一般情况下 $0 \le \int_E f dx \le \infty$。若 $\int_E f dx < \infty$，则称 f 在 E 上可积。

一个自然的问题是，Lebesgue 积分与我们所熟悉的 Riemann 积分有什么联系和区别？之后我们将详细讨论 Riemann 积分与 Lebesgue 积分的关系。这里只看一个简单的例子。设 $D(x)$ 是区间 $[0,1]$ 上的 Dirichlet 函数。即 $D(x)=\chi_{Q_0}(x)$，其中 Q_0 表示 $[0,1]$ 中的有理数的全体。根据非负简单函数积分的定义，$D(x)$ 在 $[0,1]$ 上的 Lebesgue 积分为

$$\int_0^1 D(x)dx = \int_0^1 \chi_{Q_0}(x)dx = m(Q_0) = 0 \tag{2}$$

表 1: 燃油效率与汽车价格

	(1)	(2)
燃油效率	-238.90***	-49.51
	(53.08)	(86.16)
汽车重量		1.75***
		(0.641)
常数项	11,253***	1,946
	(1,171)	(3,597)
观测数	74	74
R^2	0.220	0.293

括号内为标准误

*** p<0.01, ** p<0.05, * p<0.1

即 $D(x)$ 在 $[0,1]$ 上是 Lebesgue 可积的并且积分值为零。但 $D(x)$ 在 $[0,1]$ 上不是 Riemann 可积的。

定理 10.1 (Fubini 定理) *若 $f(x,y)$ 是 $\mathcal{R}^p \times \mathcal{R}^q$ 上的非负可测函数，则对几乎处处的 $x \in \mathcal{R}^p$，$f(x,y)$ 作为 y 的函数是 $\mathcal{R}^q$ 上的非负可测函数，$g(x) = \int_{\mathcal{R}^q} f(x,y)dy$ 是 $\mathcal{R}^p$ 上的非负可测函数。并且*

$$\int_{\mathcal{R}^p \times \mathcal{R}^q} f(x,y)dxdy = \int_{\mathcal{R}^p} \left(\int_{\mathcal{R}^q} f(x,y)dy \right) dx. \tag{3}$$

证明. Let z be some element of $xH \cap yH$. Then $z = xa$ for some $a \in H$, and $z = yb$ for some $b \in H$. If h is any element of H then $ah \in H$ and $a^{-1}h \in H$, since H is a subgroup of G. But $zh = x(ah)$ and $xh = z(a^{-1}h)$ for all $h \in H$. Therefore $zH \subset xH$ and $xH \subset zH$, and thus $xH = zH$. Similarly $yH = zH$, and thus $xH = yH$, as required.

回归分析（regression analysis) 是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。根据定理 10.1，其运用十分广泛，回归分析按照涉及的变量的多少，分为一元回归和多元回归分析；按照因变量的多少，可分为简单回归分

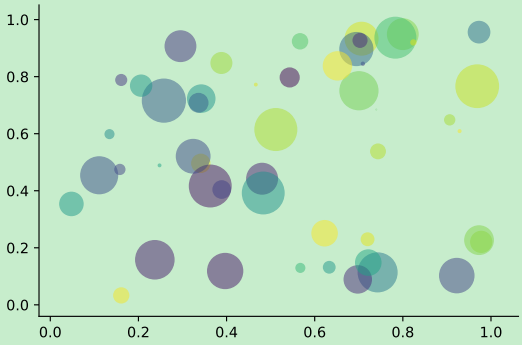

图 1: Matplotlib: Scatter Plot Example

析和多重回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。

但是由于绝对值不易作解析运算，因此，进一步用残差平方和函数来度量总偏差。偏差的平方和最小可以保证每个偏差都不会很大。于是问题归结为确定拟合函数中的常数和使残差平方和函数最小。

- Routing and resource discovery;
  - Language Models
  - Vector Space Models
- Resilient and scalable computer networks;
- Distributed storage and search.

11 示例

```
\documentclass[geye,green,pad,cn]{elegantnote}

\title{Note Example}
\author{ddswhu}
```

```
\institute{Elegant\LaTeX{} Program}
\version{1.00}
\date{\today}

\begin{document}

\maketitle

\section{Introduction}
The content of Introduction.

\end{document}
```