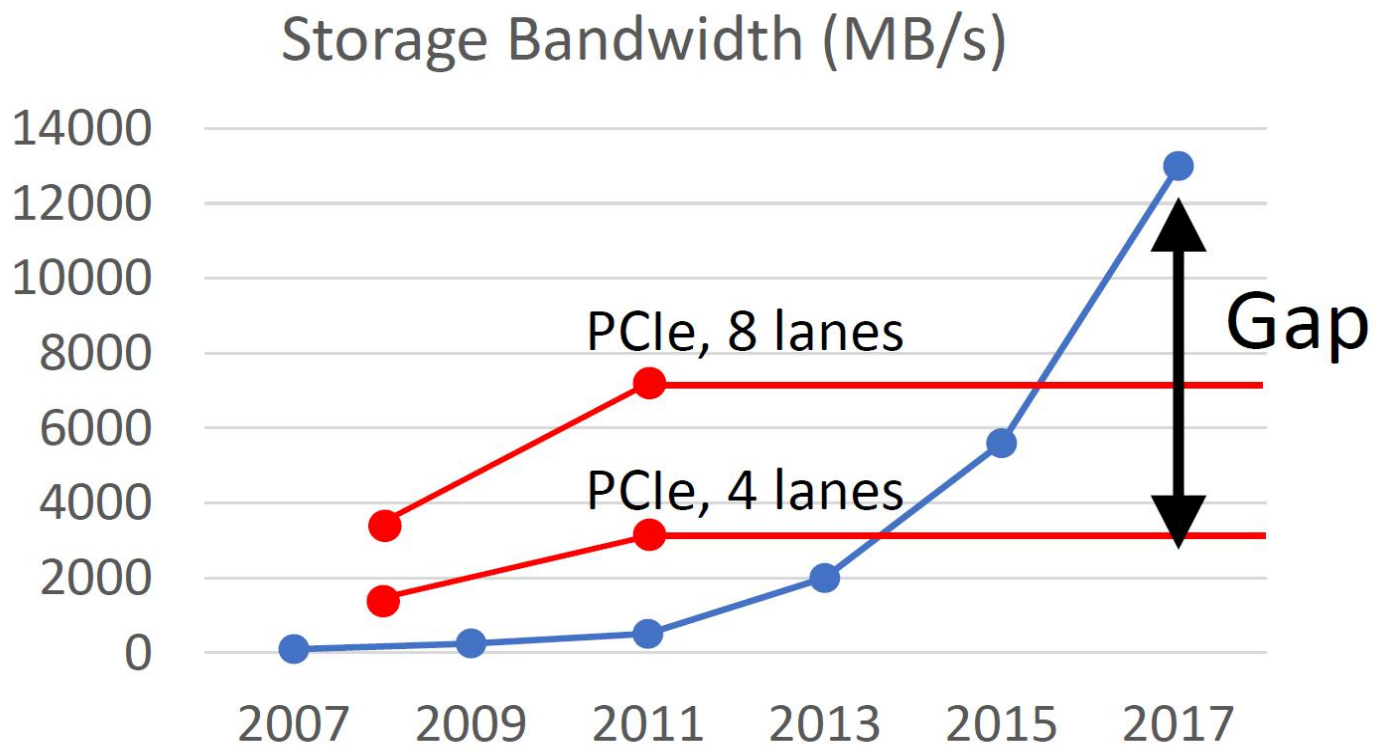
INSIDER: Designing In-Storage Computing System for Emerging High-Performance Drive

为新兴高性能驱动器设计的存储计算系统

Zain (Zhenyuan) Ruan, Tong He, Jason Cong
University of California, Los Angeles

Report By 符传杰 M201973008

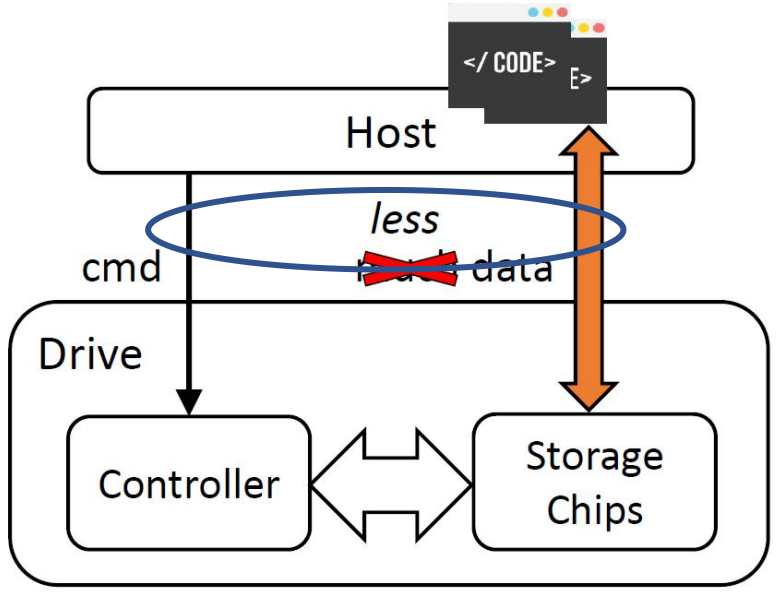
背景介绍



- 早期驱动器性能远低于接口性能
- 最新的驱动器能提供接近内存的性能
- 数据处理的瓶颈从驱动器转换到IO总线技术

In-Storage Computing(ISC)

- 基本概念
  - 主机将部分任务卸载到驱动器中，利用驱动器的性能完成计算任务。



- 可行性分析
  - 驱动器集成计算元件的成本下降
  - 驱动器的性能优于主机/驱动之间的互联性能

In-Storage Computing(ISC)

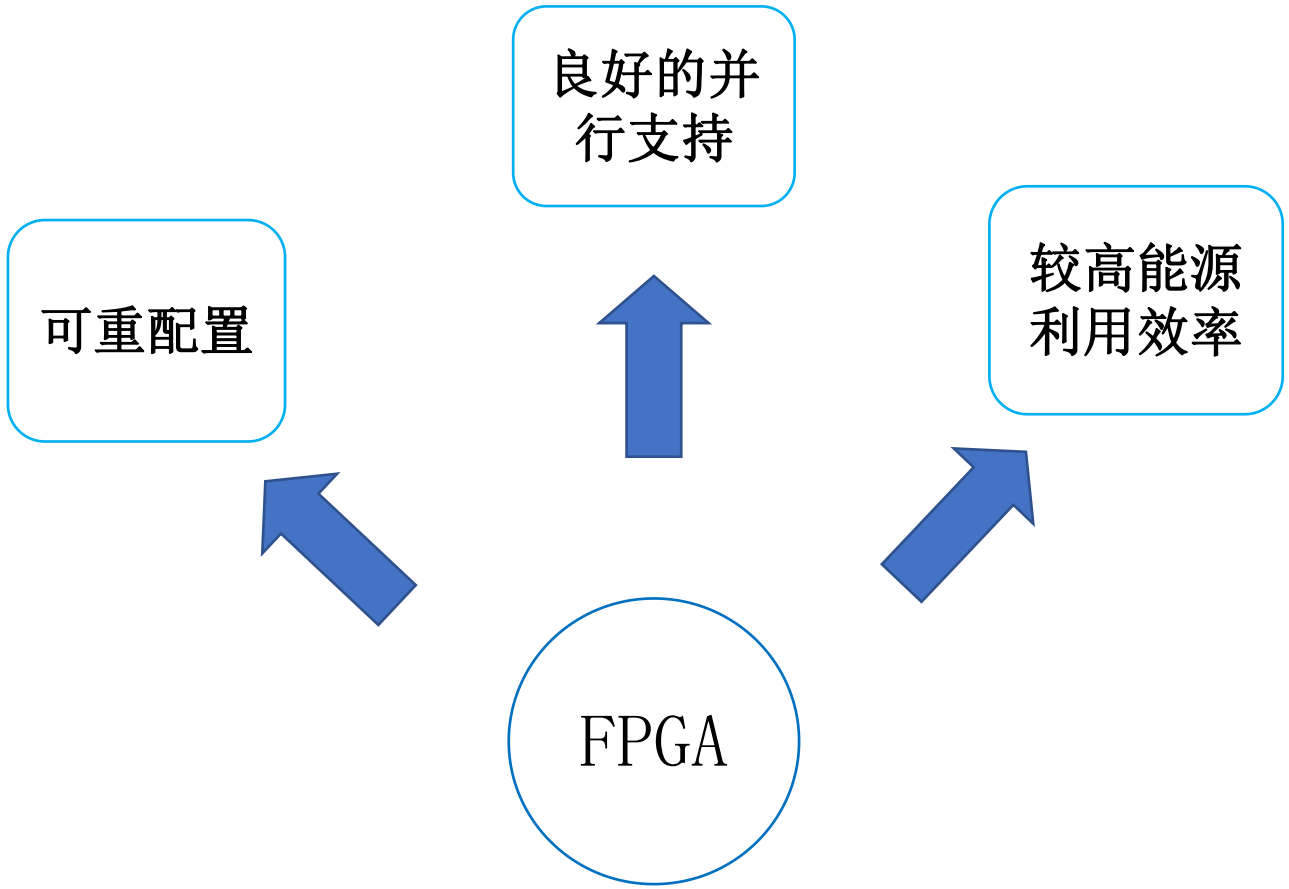
- 计算能力和扩展性的限制，如ARM和ASIC
- 缺少关键支持，如数据保护，应用调度
- 缺乏有效抽象，不兼容POSIX

性能和可扩展性

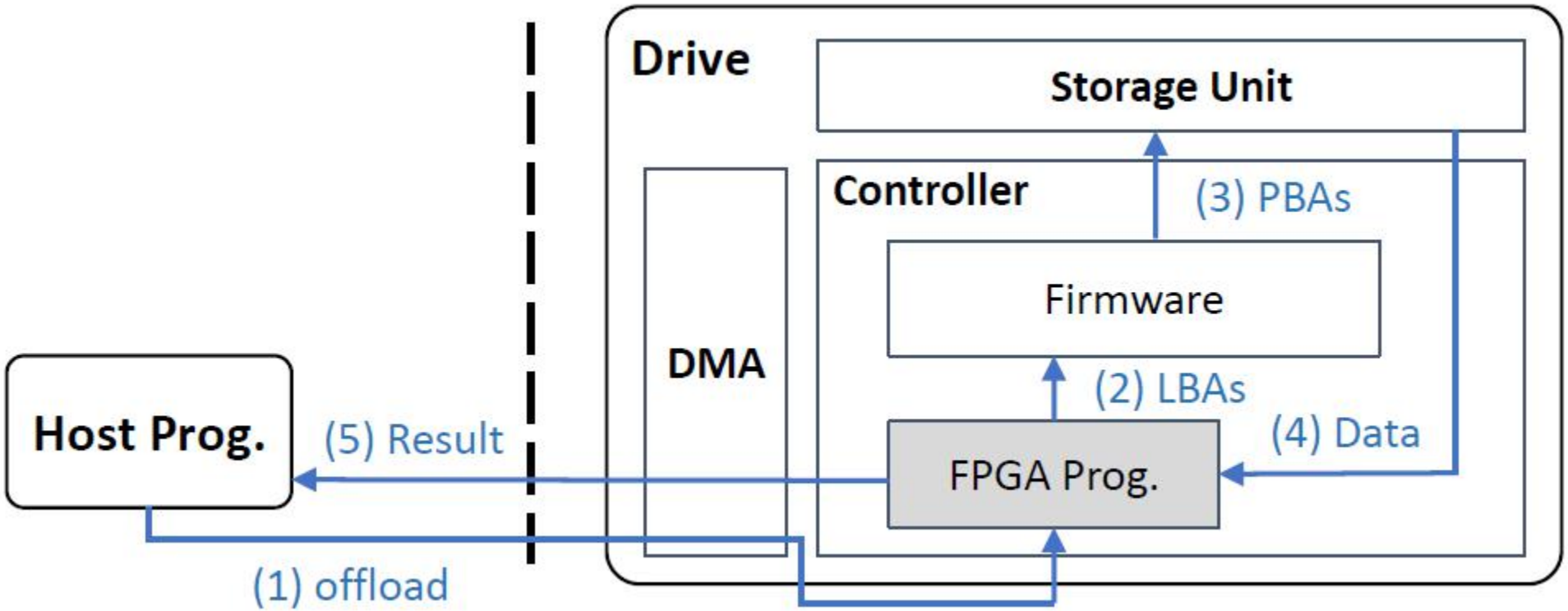
- 需求
  - 高可编程性：支持常规的ISC任务
  - 高并行：能充分利用驱动的内部带宽
  - 高能效：计算单元的加入不能显著提高驱动器能耗

	GPU	ARM	X86	ASIC	FPGA
Programmability	Good	Good	Good	No	Good
Pipeline-level parallelism	No	No	No	Best	Good
Energy efficiency	Fair	Fair	Poor	Best	Good

性能和可扩展性



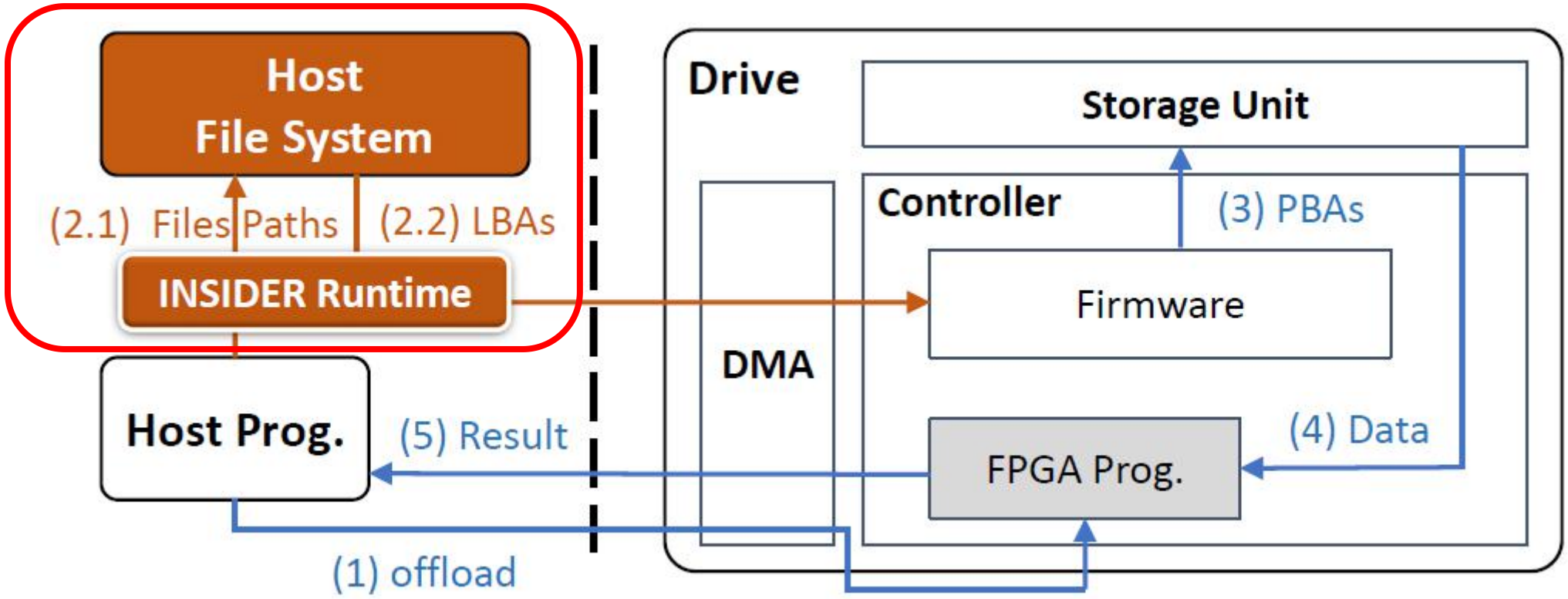
初始系统架构



缺少权限控制

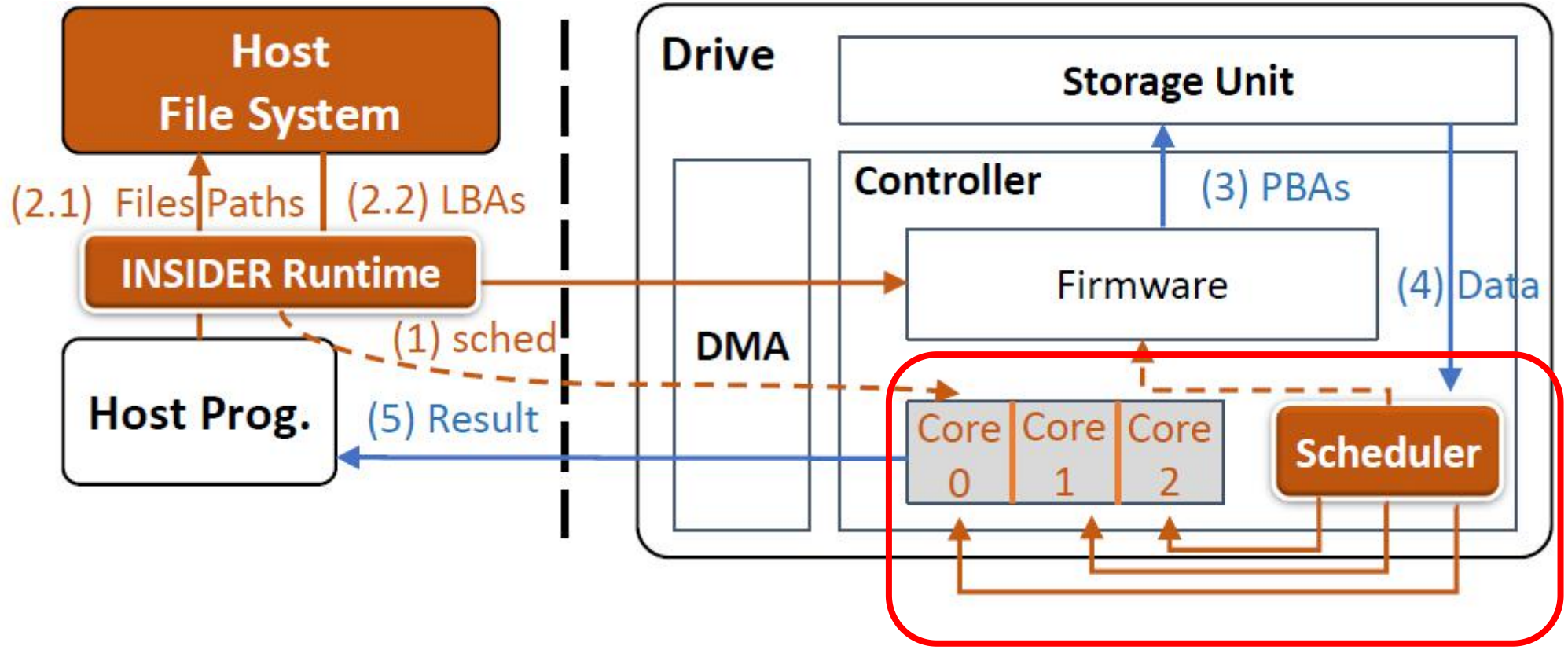
访问权限检查

- 驱动器仅负责计算
- 控制台负责处理IO请求

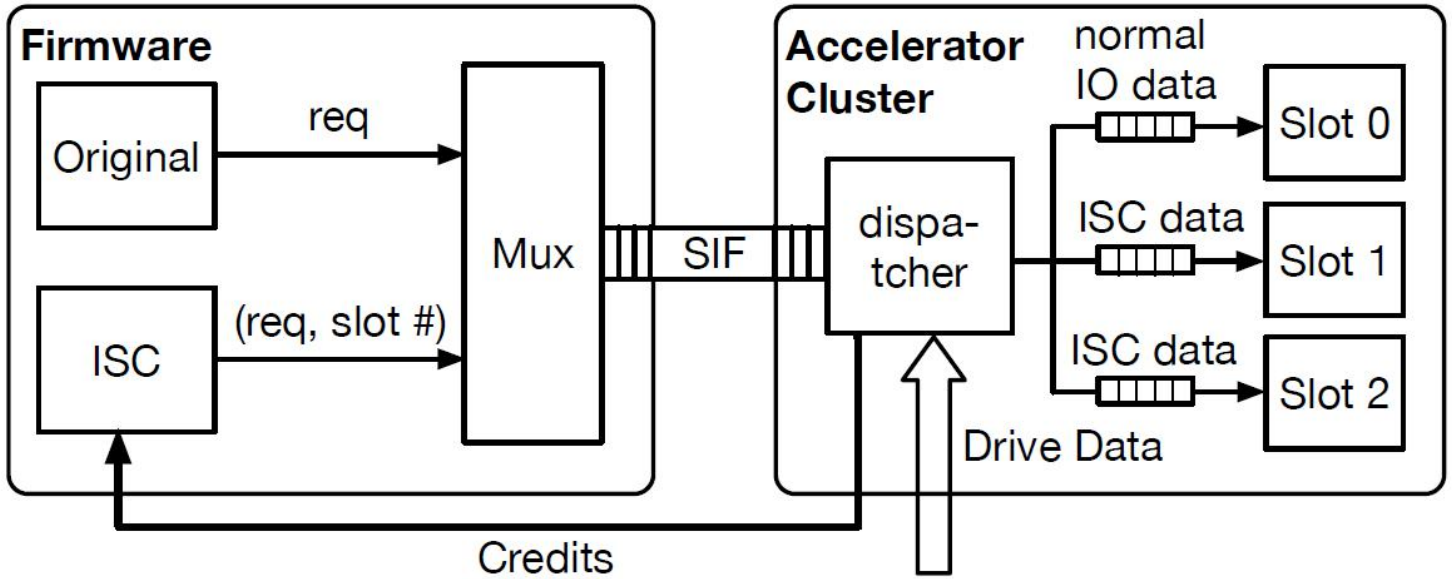


缺少对于多核FPGA的支持

多应用调度


Host
File System
(2.1) Files Paths
(2.2) LBAs
INSIDER Runtime
(1) sched
Host Prog.
(5) Result
Drive
Storage Unit
Controller
(3) PBAs
Firmware
(4) Data
DMA
Core 0
Core 1
Core 2
Scheduler

多路复用



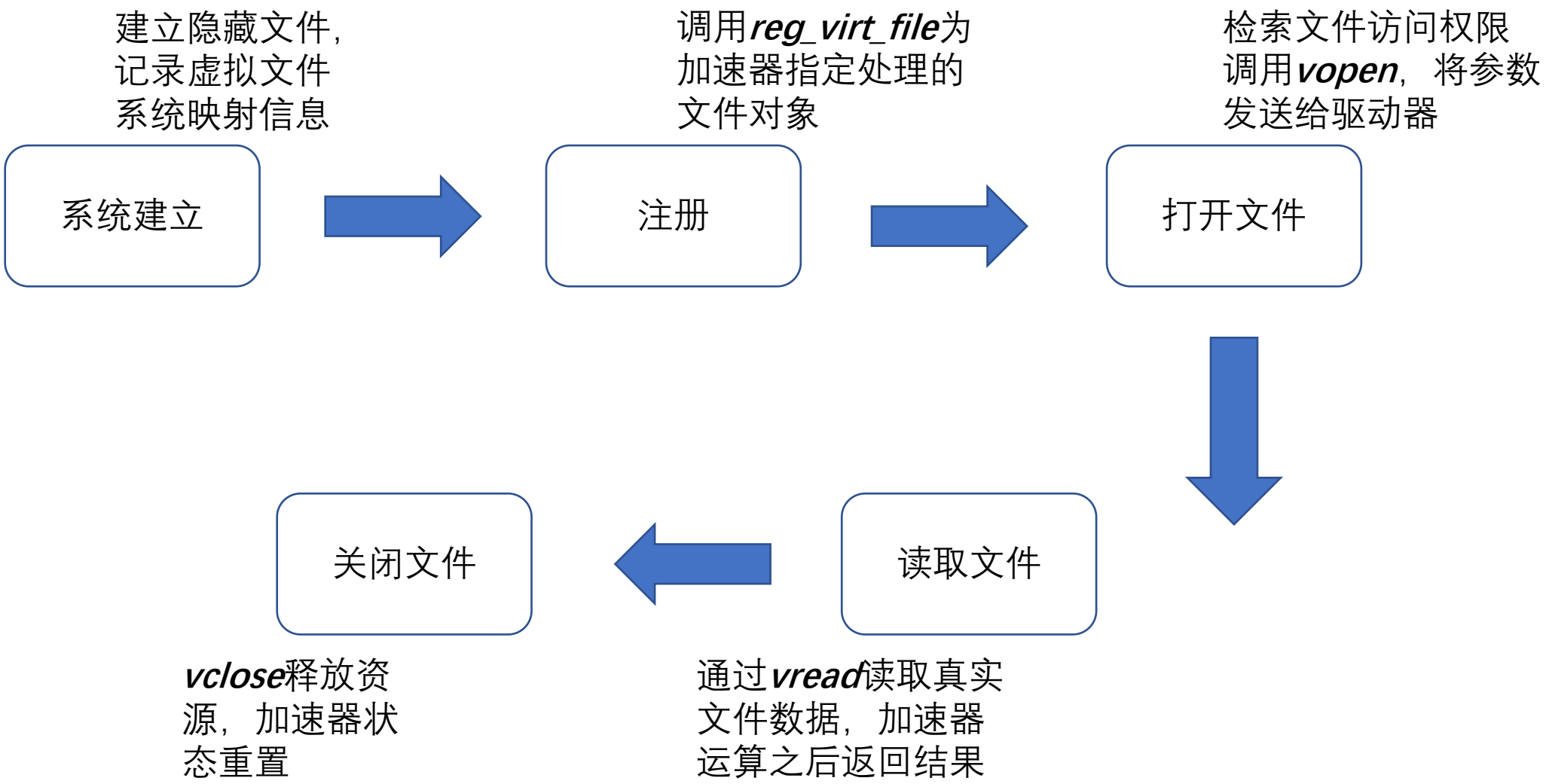
- 多路复用
  - 对固件进行扩展
  - ISC接收插槽索引
  - 多路复用器接收请求，转发至存储单元和加速器集群

模型虚拟化

- 主机端
  - 抽象ISC为简单的文件操作
  - 提供了接近POSIX的文件访问接口

1). int vopen(const char *path, int flags)
2). ssize_t vread(int fd, void *buf, size_t count)
3). ssize_t vwrite(int fd, void *buf, size_t count)
4). int vsync(int fd)
5). int vclose(int fd)
6). int vclose(int fd, size_t *rfile_written_bytes)
7). string reg_virt_file(string file_path, string acc_id)
8). string reg_virt_file(tuple<string, uint, uint> file_sg_list, string acc_id)
9). bool send_params(int fd, void *buf, size_t count)

示例


建立隐藏文件，
记录虚拟文件
系统映射信息
系统建立
调用*reg_virt_file*为
加速器指定处理的
文件对象
注册
检索文件访问权限
调用*vopen*，将参数
发送给驱动器
打开文件
读取文件
通过*vread*读取真实
文件数据，加速器
运算之后返回结果
关闭文件
*vclose*释放资
源，加速器状
态重置

驱动端

- 驱动端维护三个队列：输入队列，输出队列，参数队列

- 先从参数队列读取两个参数，即输入队列读取的范围；接着从输入队列读取数据，完成计算任务后写入输出队列

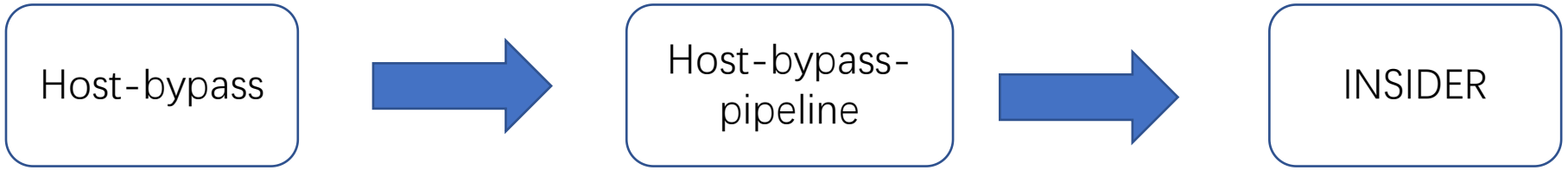
驱动端只需要关注计算逻辑

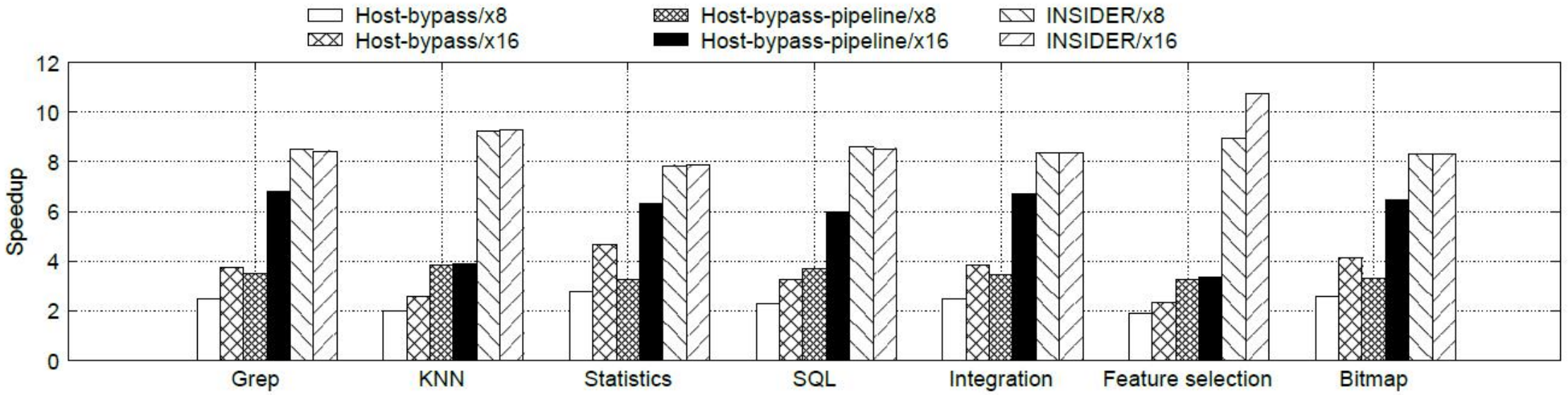
实验条件

- 常见应用及其描述

应用	描述	任务分配
Grep	字符串匹配；	卸载全部任务至驱动端。
KNN	K近邻算法；	卸载距离计算任务到驱动端。
位图压缩	——	卸载游程编码任务到驱动端。
Statistics	按行执行统计；	卸载数据去重任务到驱动端。
SQL查询	包括：select，sum，where这些操作；	卸载过滤相关任务到驱动端。
Integration	合并不同数据源的数据；	卸载全部任务至驱动。
Feature Selection	特征筛选；	卸载全部任务至驱动。

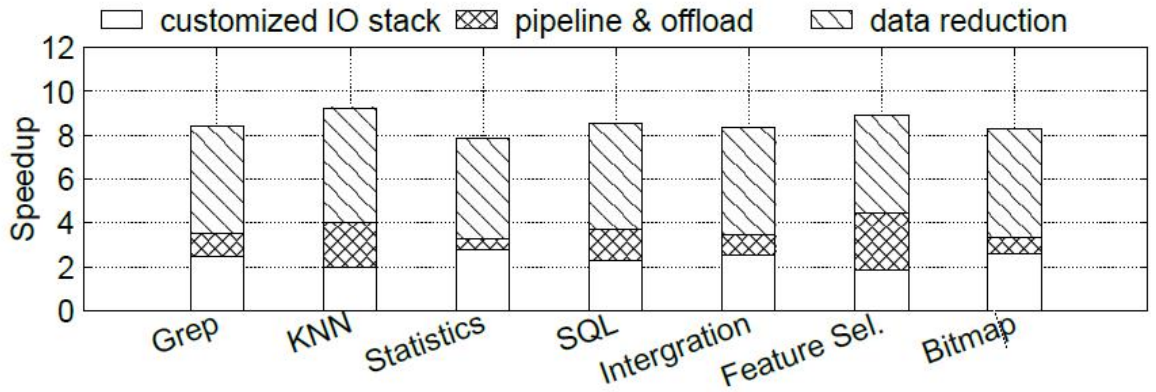
拆解加速过程


Host-bypass
Host-bypass-pipeline
INSIDER

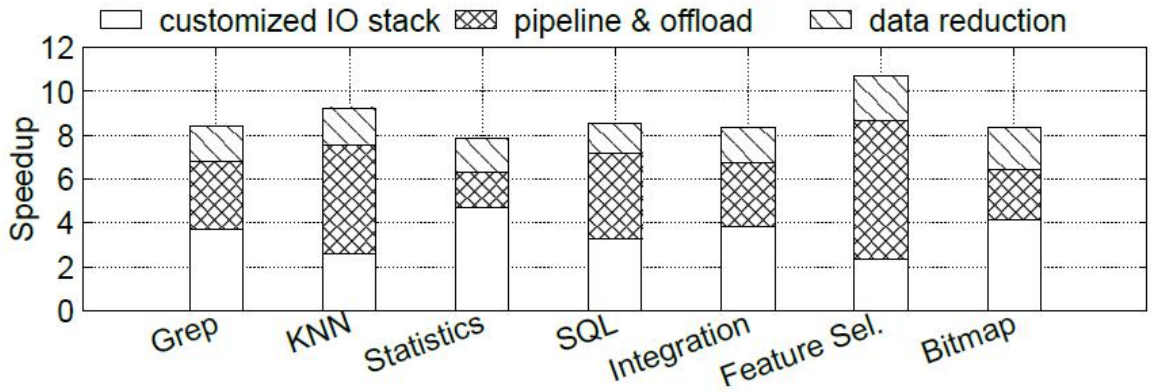

Host-bypass/x8
Host-bypass/x16
Host-bypass-pipeline/x8
Host-bypass-pipeline/x16
INSIDER/x8
INSIDER/x16
Speedup
0
2
4
6
8
10
12
Grep
KNN
Statistics
SQL
Integration
Feature selection
Bitmap

拆解加速过程

- 定制IO堆栈
- 流水线级别并行
- 减少数据量



(a) INSIDER*/x8* (*i.e,* the bus-limited case).

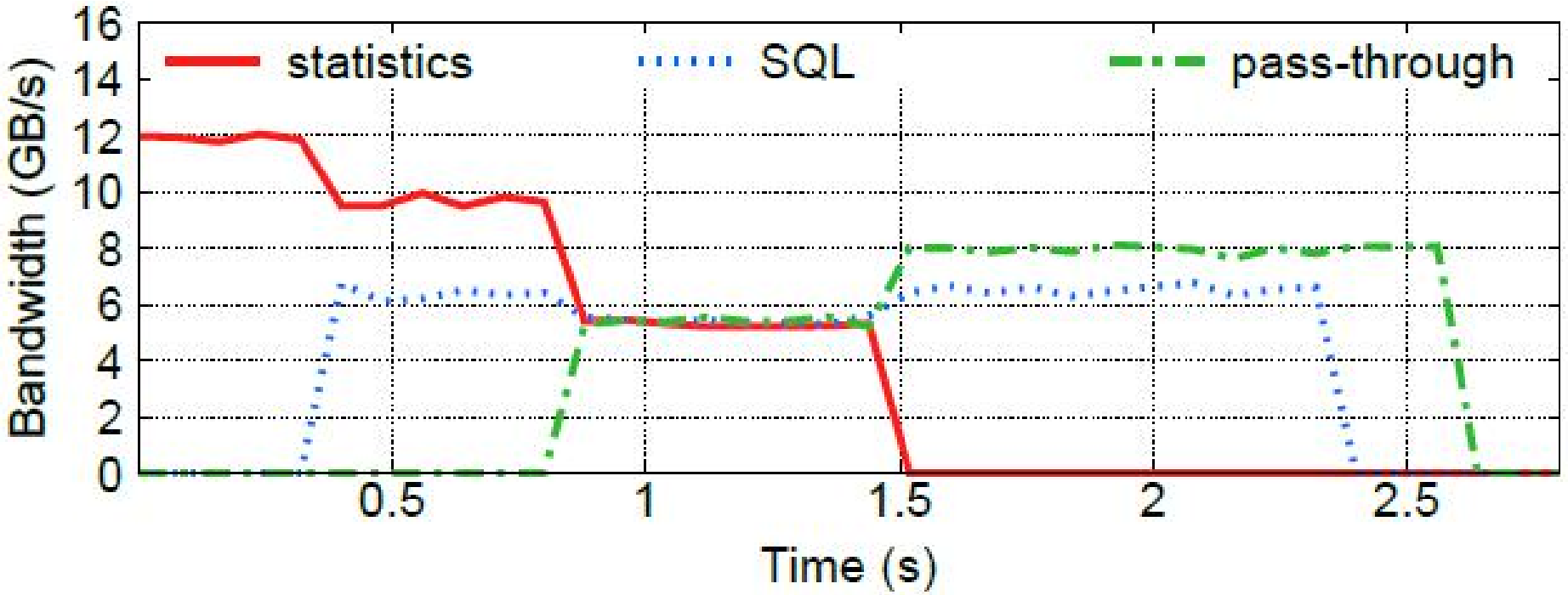


(b) INSIDER*/x16* (*i.e,* the bus-ample case).

瓶颈分析

	Host-bypass/x8	*Host-bypass/x16*	INSIDER/*x8*	INSIDER/*x16*
Grep	PCIe	PCIe	Drive	Drive
KNN	PCIe	Comp.	Drive	Drive
Statistics	PCIe	PCIe	Drive	Drive
SQL query	PCIe	Comp.	Comp.	Comp.
Integration	PCIe	PCIe	Drive	Drive
Feature selection	Comp.	Comp.	PCIe	Drive
Bitmap decompression	PCIe	PCIe	Drive	Drive

带宽利用率


statistics
SQL
pass-through
Bandwidth (GB/s)
16
14
12
10
8
6
4
2
0
0.5
1
1.5
2
2.5
Time (s)

开发时间分析

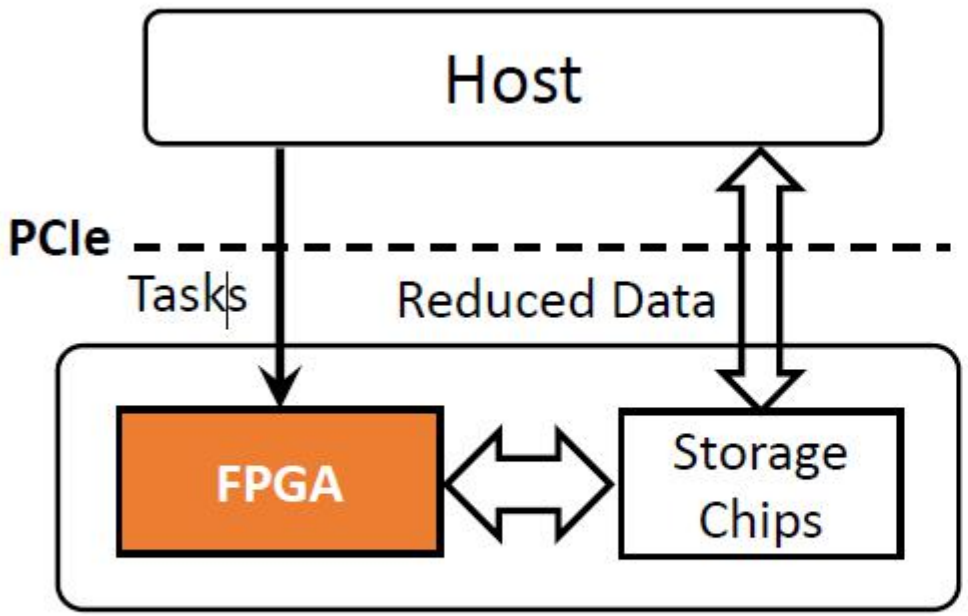
应用	开发时间	主机端代码量	驱动端代码量
Grep	3	51	193
KNN	2	77	72
位图压缩	4	94	145
Statistics	3	65	170
SQL查询	5	97	256
Integration	5	41	307
Feature Selection	9	50	632

开发时间对比

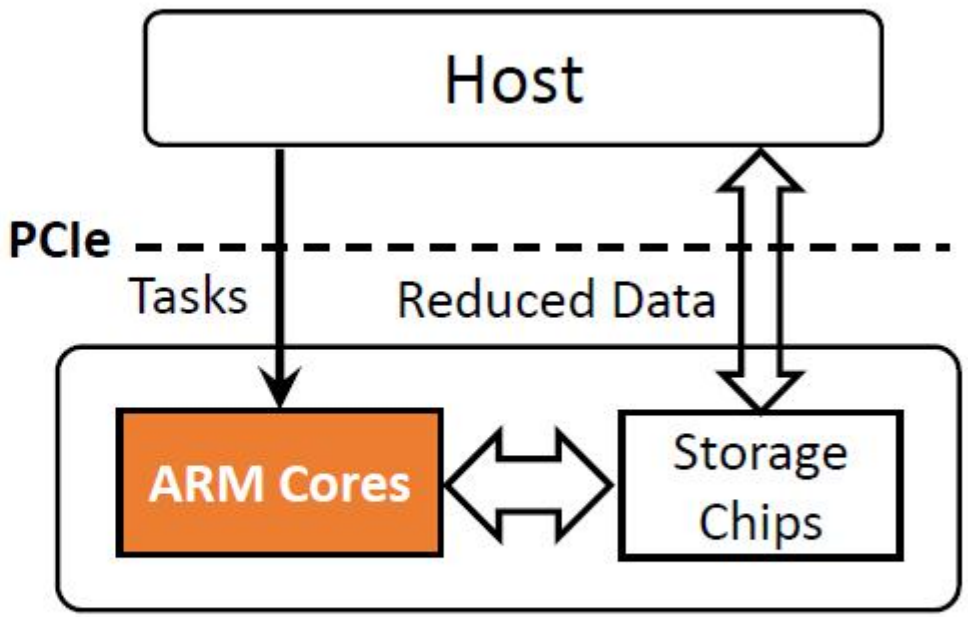
Description	Name	LOC (C)	Devel. Time (Person-months)
Simple IO operations [7]	`Base-IO`	1500	1
Virtualized SSD interface with OS bypass and permission checking [8]	`Direct-IO`	1524	1.2
Atomic writes tailored for scalable database systems based on [10]	`Atomic-Write`	901	1
Direct-access caching device with hardware support for dirty data tracking [5]	`Caching`	728	1
SSD acceleration for MemcacheDB [9]	`Key-Value`	834	1
Offload file appends to the SSD	`Append`	1588	1

INSIDER VS ARM-ISC

➢架构

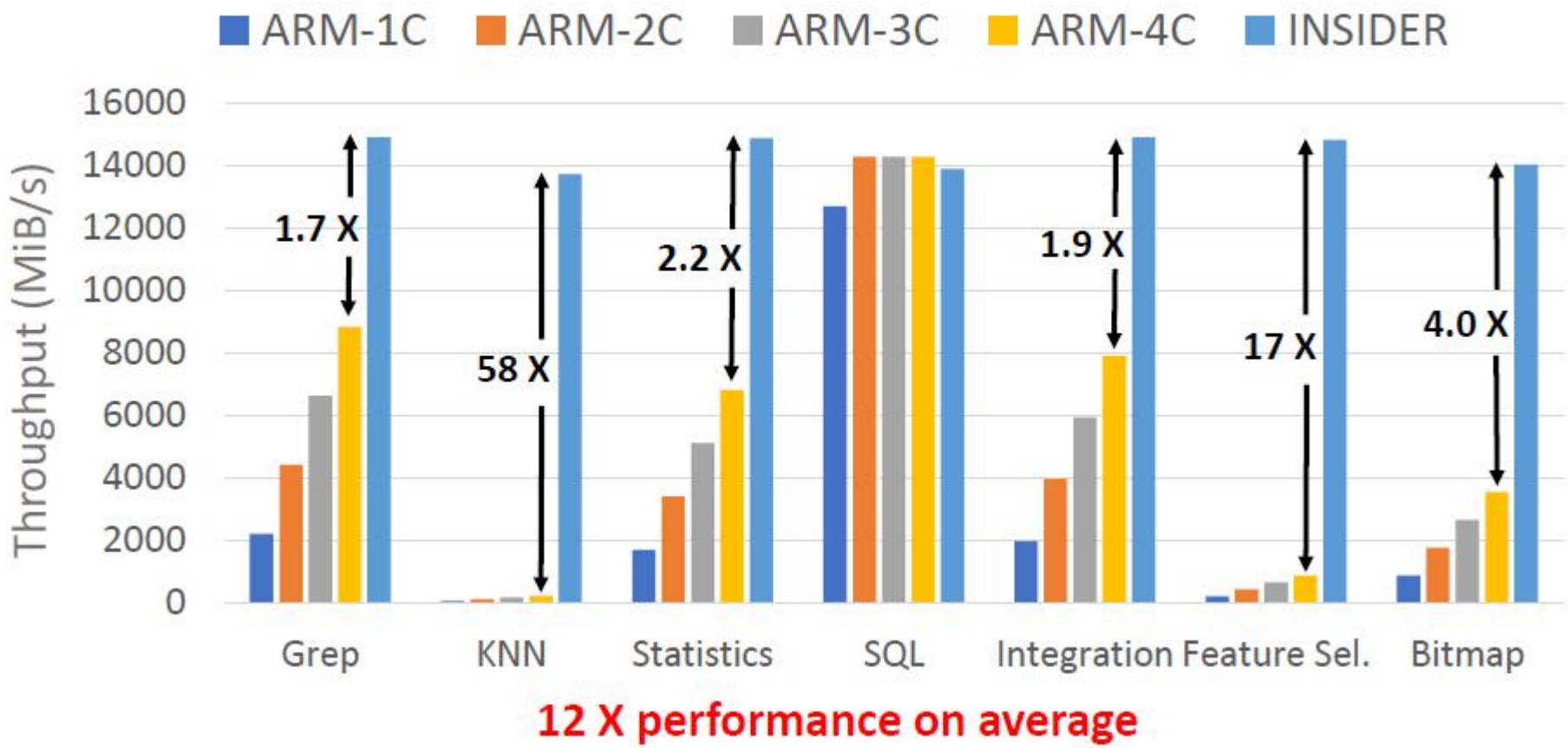


INSIDER (Xilinx Virtex / Artix)



ARM-ISC (Cortex-A72)

带宽


ARM-1C
ARM-2C
ARM-3C
ARM-4C
INSIDER
Throughput (MiB/s)
16000
14000
12000
10000
8000
6000
4000
2000
0
1.7 X
58 X
2.2 X
1.9 X
17 X
4.0 X
Grep
KNN
Statistics
SQL
Integration
Feature Sel.
Bitmap
12 X performance on average

总结

- 贡献
  - 实现高效的性能和可扩展性
  - 提供了共享环境下的访问控制和资源调度
  - 对ISC进行简单抽象

- 不足
  - 资源调度没有对于请求的优先级进行区分
  - 文章没有涉及复杂场景下的任务卸载

感谢聆听！

Report By 符传杰 M201973008