Center for Domain-Specific Computing

CFAR

When Apache Spark Meets FPGAs:
A Case Study for Next-Generation DNA Sequencing Acceleration

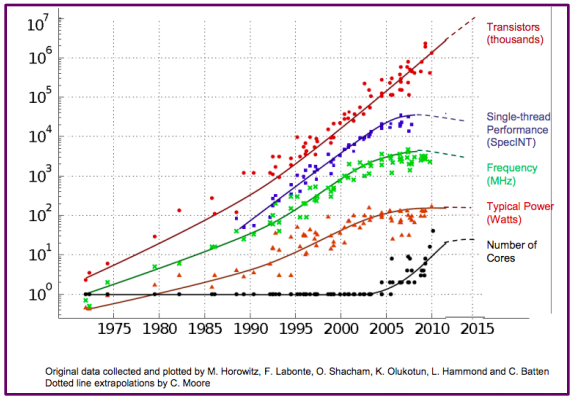
Yu-Ting Chen, Jason Cong, Zhenman Fang, Jie Lei and Peng Wei

University of California, Los Angeles

UCLA COMPUTER SCIENCE DEPARTMENT

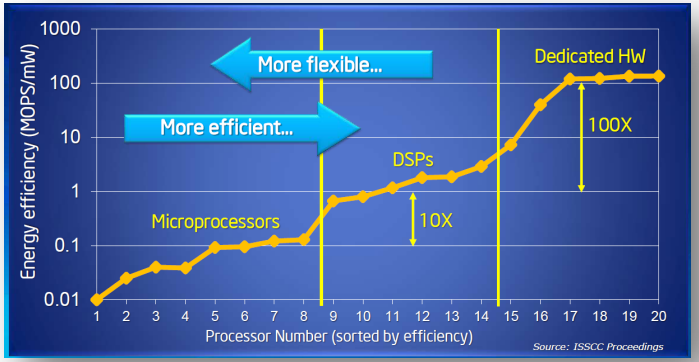
We are forced to explore heterogeneity

- **When Dennard scaling comes to the end and Moore's law slows down as components reach atomic scale**
  - **Shift from single-core to multi-core**
  - **Take the pain to learn parallel programming**
    - **Memory ordering, locking, load balancing, …**
  - **Even homogeneous multi-core architectures are not able to drive continued perf. and energy improvement that we have come to expect in the past**

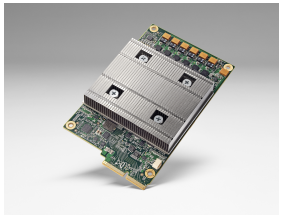


Heterogeneous Architecture

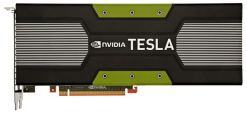
- **Why heterogeneity?**
  - Customized accelerators promise orders-of-magnitude perf./watt gains
- **Accelerator-centric architectures**
  - Application-specific functional units
  - Domain-specific building blocks
  - Reconfigurable/reprogrammable logic
    - GPUs, CGRAs, FPGAs, …



Source: Bob Broderson, Berkeley Wireless group



Tensor Processing Unit
Source: Google



GPGPU
Source: NVIDIA

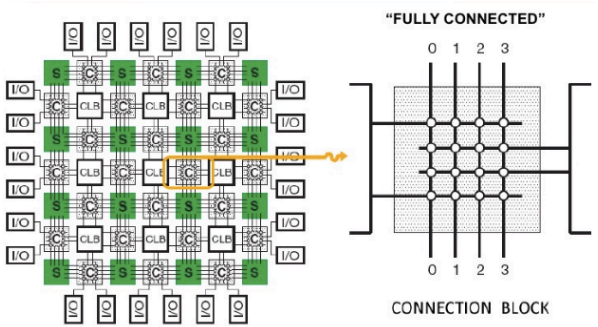
FPGA
Source: Alpha Data

What is an FPGA?

- **Field Programmable Gate Array (FPGA)**
  - Reconfigurable hardware
  - Can be used to solve any problem which is computable
    - Not always fast, though
  - Is well-known to be used for ASIC prototyping and hardware emulation
    - Prototype accelerators
  - Can also be directly used as a compute accelerator

Xilinx Virtex-7 FPGA
Source: Xilinx

Altera Stratix-10 FPGA
Source: Altera



http://www.slideshare.net/omutukuda/presentation-1993175

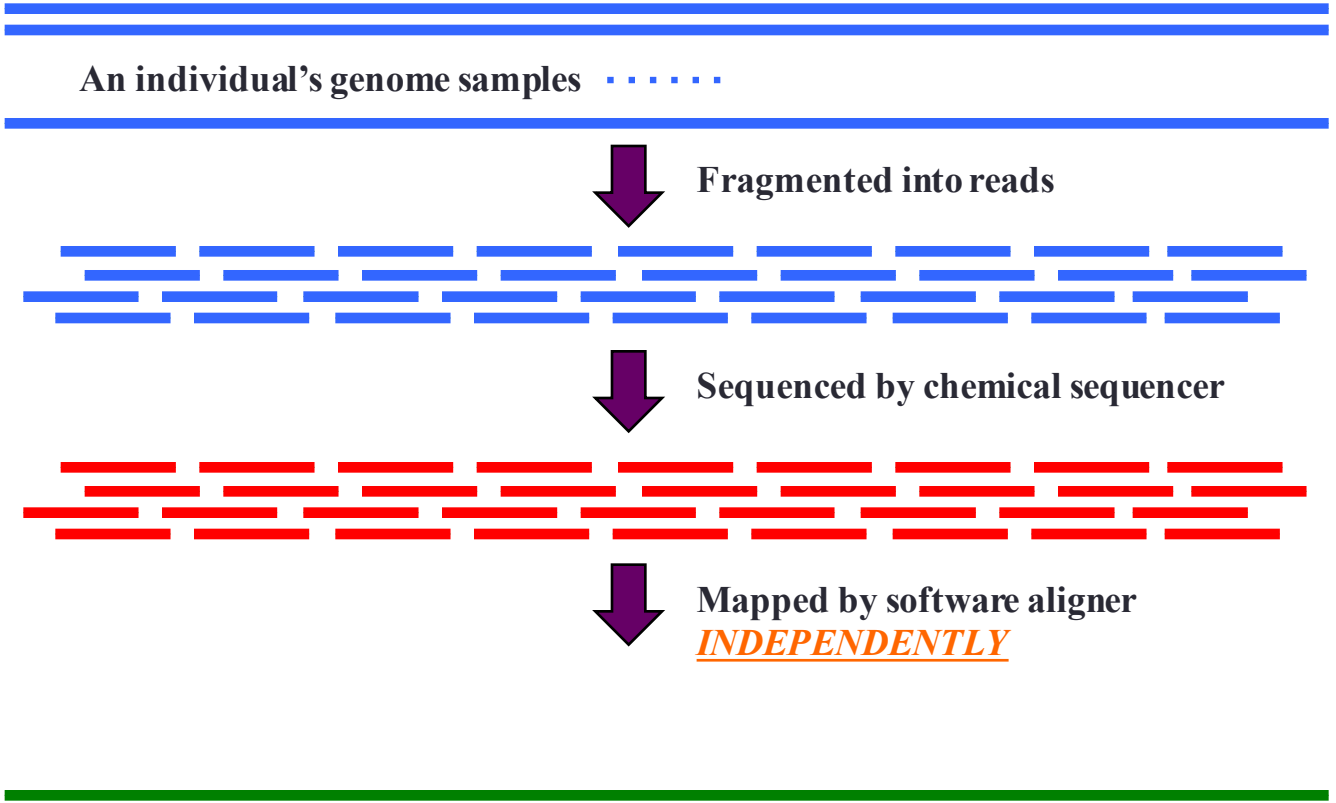
Why FPGAs in datacenter?

- **FPGAs are reconfigurable**
  - You do not want to be as extravagant as Google to tape-out an ASIC chip only for one app
  - Not many applications are so important as to be equipped with a customized chip
- **FPGAs are energy-efficient**
  - A high-end FPGA board typically consumes ~20 watt
- **Concrete examples have demonstrated harnessing FPGAs in datacenter**
  - Microsoft Catapult for the Bing search engine: 2x speedup with 10% more power consumption
- **An FPGA fabric is (probably) going to be a "free lunch" in the near future**
  - Altera, now part of Intel => An FPGA fabric will probably be a default component in a server
- **CPU-FPGA systems are not fully elucidated**
  - So many research opportunities, so much potential

The Objectives of This Case Study

- **Accelerating an application that is a matter of life or death**
  - **DNA sequencing is expected to be widely used in precision medicine, like cancer treatment**
  - **A cancer cell's genome mutates probably *in a few days***
  - **The state-of-the-art sequencing pipeline typically takes *a week or so* to sequence a patient's genome, which, for clinic use, is too long**
- **Proposing an approach to efficient integration of FPGAs into Apache Spark**
- **Presenting what issues need to be addressed to operate FPGAs at scale**
- **Attracting more attention from system architects to make better CPU-FPGA systems**

Next-Generation DNA Sequencing



The Solution: Heterogeneous Cluster Computing

- **Processing billions of reads (strings) independently**
  - **Fit the MapReduce programming model perfectly**
  - **As for the long reference genome? Spark's broadcast variables**
- **Inside each read's alignment process**
  - **Step #1: Seeding**
    - **Exact string matching**
    - **Linear time complexity**
  - **Step #2: Extending**
    - **Approximate string matching**
    - **The Smith-Waterman dynamic programming algorithm (Quadratic time complexity)**
    - **Accelerated by FPGAs**

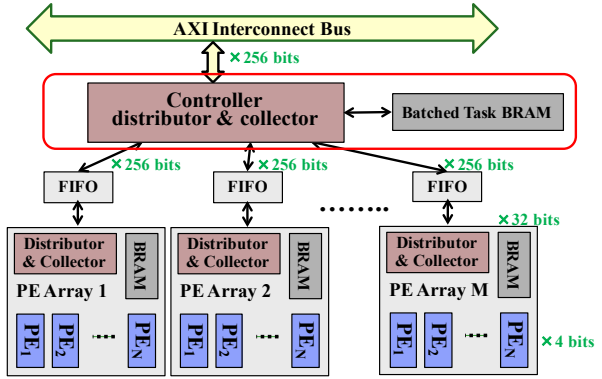
Straightforward Integration: 1+1 < 0.001

The Spark Program

- CS-BWAMEM [HitSeq `15]
- Aligning billions of short reads onto the reference human genome in parallel

The Accelerator [FCCM `15]

- A throughput-oriented FPGA accelerator for the Smith-Waterman DP kernel



The Straightforward JNI Integration

- CPU: 2.1 x 10^3 reads per second
- FPGA: 1.6 reads per second

On CPU

One read
⇒ 24 DPs
⇒ 20 µs per DP
⇒ 2.1 x 10^3 reads/s

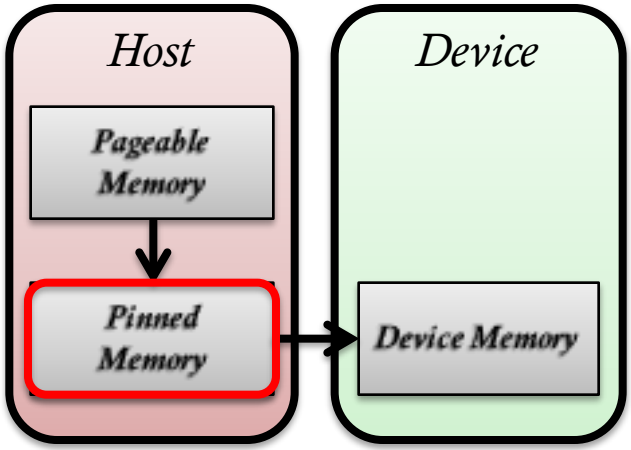
On FPGA

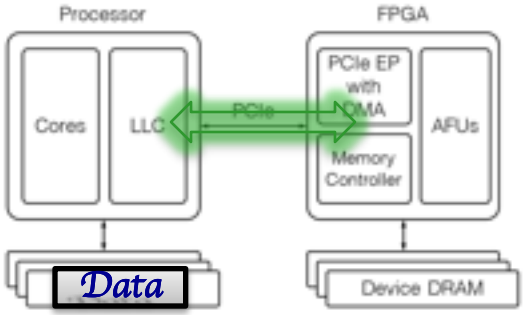
One DP
⇒ *25 ms data transfer*
⇒ 1.6 reads/s

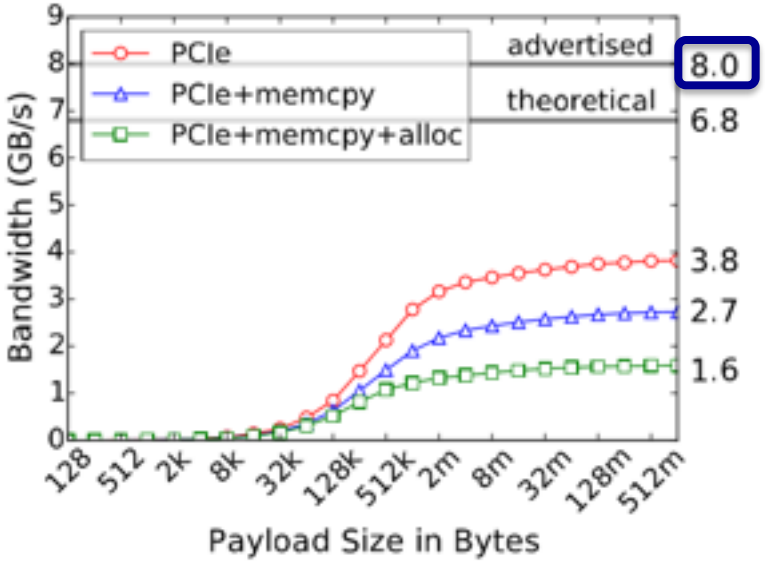
While *JNI* serves as a standard approach to connect JVMs with FPGAs, a straightforward integration through JNI degrades the performance by ***1000x***.

What happened in a CPU-FPGA communication instance?

- Java Heap ⇔ Native Memory
- Host Memory ⇔ Device Memory







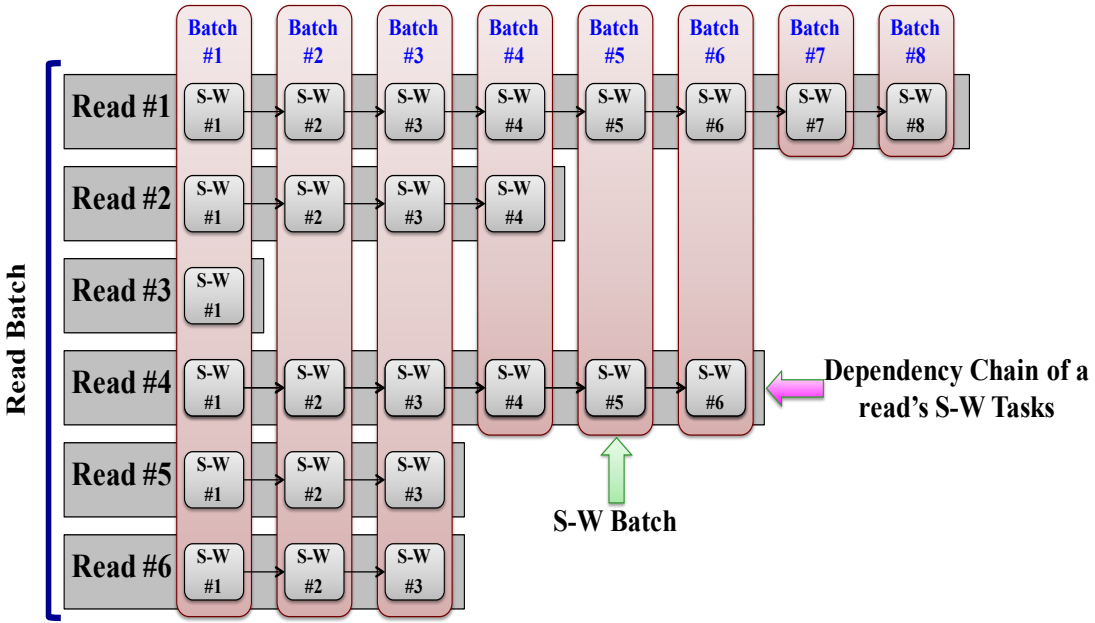
Why communication matters?

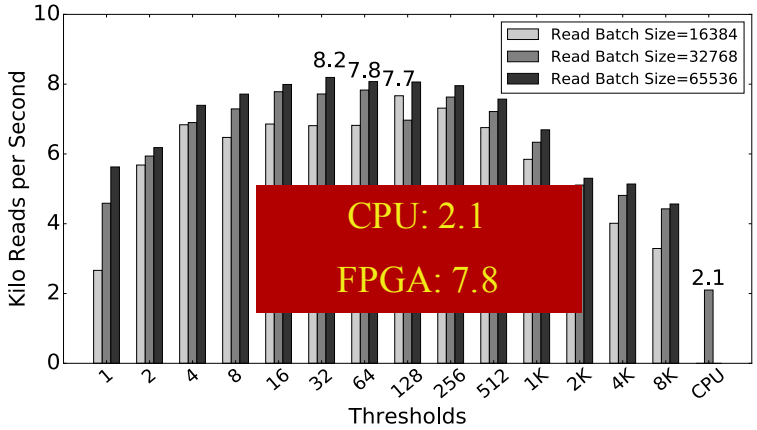
- Each map function is likely to process only a small volume of data with a small amount of execution time
  - One read is only 101 ASCII characters
  - One line of a text file
  - One record of a NoSQL table
  - …
- Communication overhead can be amortized by batch processing

```
def map_func(input:U):V = {
  // U => P => Q => V
  t1:P = cnv1(input)
  t2:Q = cnv2(t1)
  t3:V = cnv3(t2)
  t3
}
rdd_out = rdd_in.map(ele=>map_func(ele))
```

```
def map_func(input:Array[U]):Array[V] = {
  // Array[U] => … => Array[V]
  t1:Array[P] = cnv1_batch(input)
  t2:Array[Q] = cnv2_batch(t1)
  t3:Array[V] = cnv3_batch(t2)
  t3
}
rdd_out = rdd_in.map(ele=>map_func(ele))
```

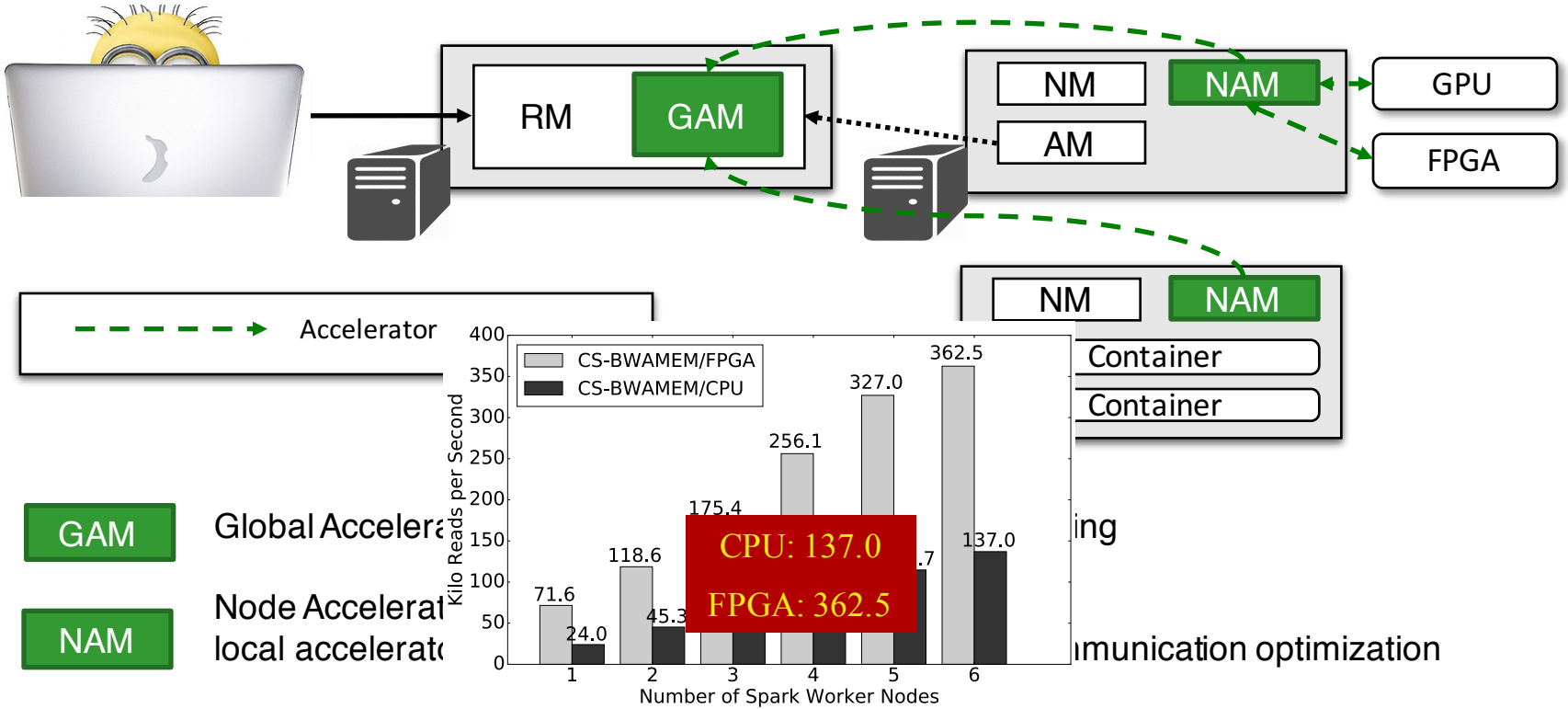
Let's first do batch processing manually





Dependency/Irregularity-Aware Batch Processing

Accelerator-as-a-Service



Source: https://spark-summit.org/2016/events/deploying-accelerators-at-datacenter-scale-using-spark/

Lessons Learned and Open Discussion

- From homogeneous to heterogeneous => requirement of corresponding systems
  - lightweight, unified and efficient processor-accelerator communication
- While various FPGA-based accelerators have been proposed, CPU-FPGA systems are being built on top of "naive" communication stack
  - Pageable memory => pinned memory => device memory
- MapReduce does not take accelerators into consideration, and does not seem to be accelerator-friendly
  - A map function often has little execution time, but considerable aggregate time
  - It must be greatly helpful if automatic code transformation for batch processing works
- A generic accelerator-aware big-data framework is needed, even if not now

Acknowledgement

THANKS FOR YOUR ATTENTION.