Customizable Computing at Datacenter Scale

Jason Cong

Chancellor's Professor, UCLA

Director, Center for Domain-Specific Computing

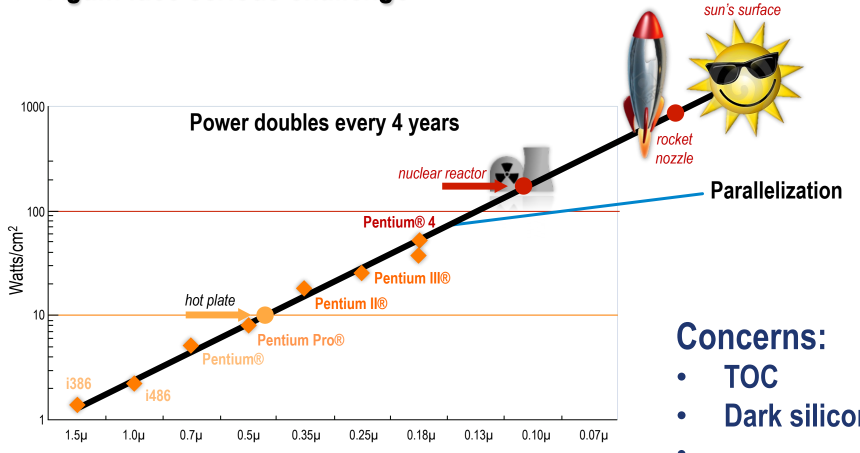
cong@cs.ucla.edu

http://cadlab.cs.ucla.edu/~cong

1

Challenge with Processor Design – Power Barrier

- Current solution: *Parallelization*
- Again face serious challenge

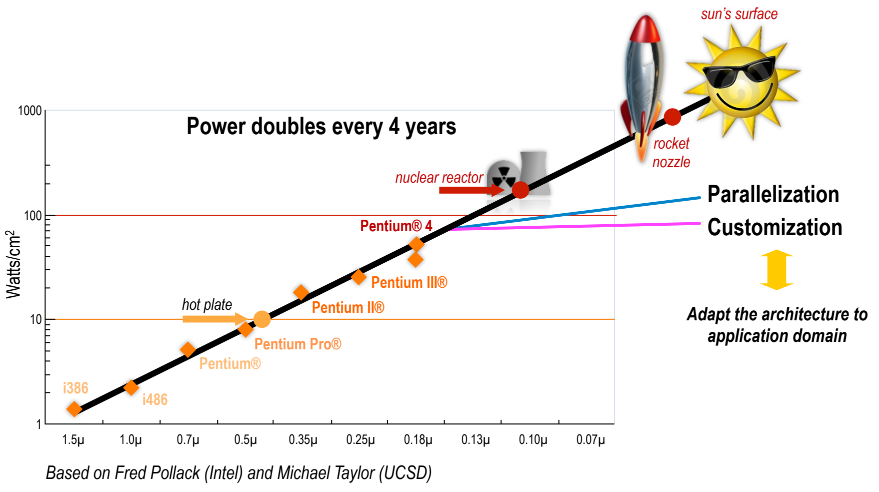


Based on Fred Pollack (Intel) and Michael Taylor (UCSD)

Concerns:
- TOC
- Dark silicon
- …

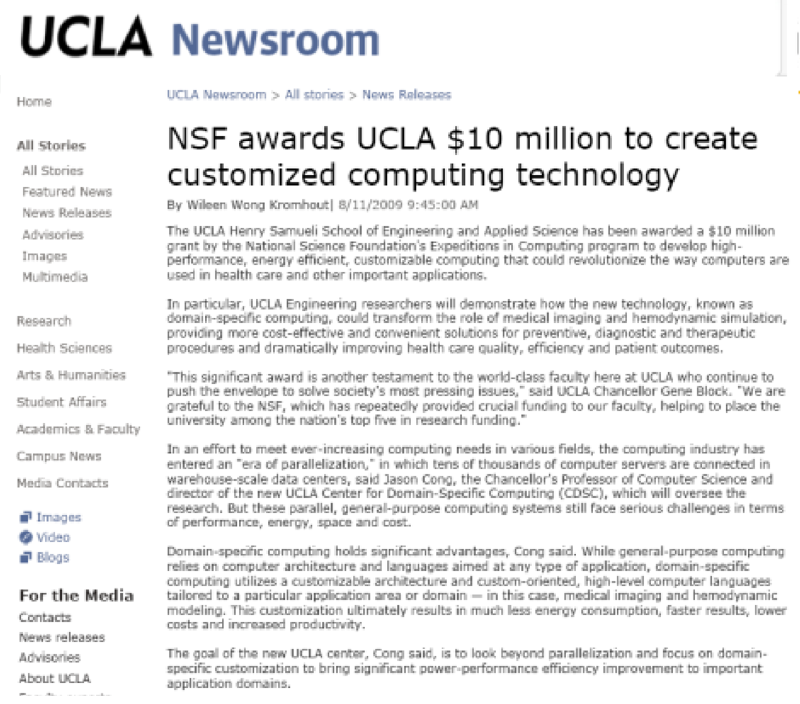
2

CDSC Focus: Customization and Specialization



Based on Fred Pollack (Intel) and Michael Taylor (UCSD)

3



NSF awards UCLA $10 million to create customized computing technology

By Wileen Wong Kromhout| 8/11/2009 9:45:00 AM

The UCLA Henry Samueli School of Engineering and Applied Science has been awarded a $10 million grant by the National Science Foundation's Expeditions in Computing program to develop high-performance, energy efficient, customizable computing that could revolutionize the way computers are used in health care and other important applications.

In particular, UCLA Engineering researchers will demonstrate how the new technology, known as domain-specific computing, could transform the role of medical imaging and hemodynamic simulation, providing more cost-effective and convenient solutions for preventive, diagnostic and therapeutic procedures and dramatically improving health care quality, efficiency and patient outcomes.

"This significant award is another testament to the world-class faculty here at UCLA who continue to push the envelope to solve society's most pressing issues," said UCLA Chancellor Gene Block. "We are grateful to the NSF, which has repeatedly provided crucial funding to our faculty, helping to place the university among the nation's top five in research funding."

In an effort to meet ever-increasing computing needs in various fields, the computing industry has entered an "era of parallelization," in which tens of thousands of computer servers are connected in warehouse-scale data centers, said Jason Cong, the Chancellor's Professor of Computer Science and director of the new UCLA Center for Domain-Specific Computing (CDSC), which will oversee the research. But these parallel, general-purpose computing systems still face serious challenges in terms of performance, energy, space and cost.

Domain-specific computing holds significant advantages, Cong said. While general-purpose computing relies on computer architecture and languages aimed at any type of application, domain-specific computing utilizes a customizable architecture and custom-oriented, high-level computer languages tailored to a particular application area or domain — in this case, medical imaging and hemodynamic modeling. This customization ultimately results in much less energy consumption, faster results, lower costs and increased productivity.

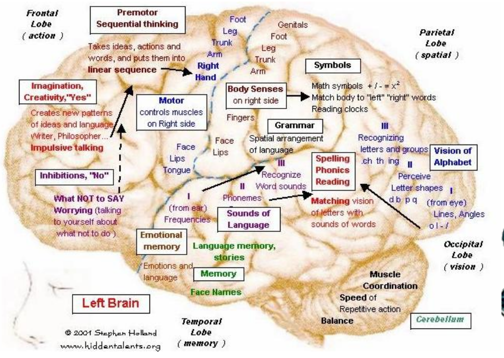
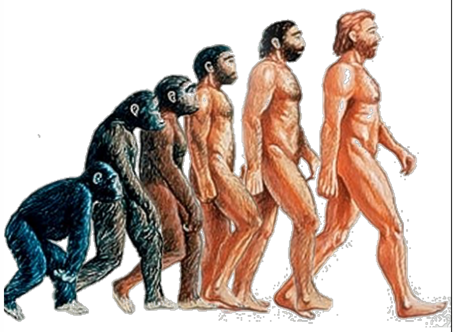
The goal of the new UCLA center, Cong said, is to look beyond parallelization and focus on domain-specific customization to bring significant power-performance efficiency improvement to important application domains.

Overview of Our Approach -- Customized Computing with Accelerator-Rich Architectures

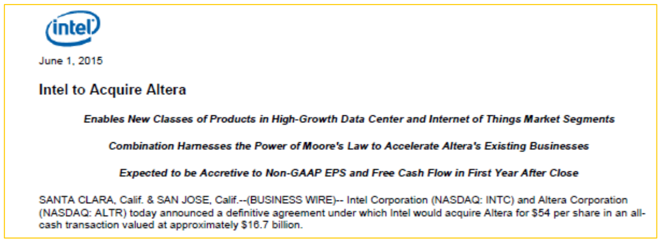
- **Extensive use of dedicated and composable accelerators**
  - Most computations are carried on accelerators – not on processors!
- A fundamental departure from von Neumann architecture
- Why now?
  - Previous architectures are device/transistor limited
  - Von Neumann architecture allows maximum device reuse
    - One pipeline serves all functions, fully utilized
- Future architectures
  - Plenty of transistors, but power/energy limited (dark silicon)
  - Customization and specialization for maximum energy efficiency
- A story of specialization

Lessons from Nature: Human Brain and Advance of Civilization

- High power efficiency (20W) of human brain comes from specialization
  - Different region responsible for different functions
- Remarkable advancement of civilization also from specialization
  - More advanced societies have higher degree of specialization


Frontal Lobe (action)
Premotor Sequential thinking
Takes ideas, actions and words, and puts them into linear sequence
Right Hand
Foot
Leg
Trunk
Arm
Genitals
Foot
Leg
Trunk
Arm
Parietal Lobe (spatial)
Symbols
Imagination, Creativity,"Yes"
Creates new patterns of ideas and language Writer, Philosopher...
Impulsive talking
Motor controls muscles on Right side
Body Senses on right side
Math symbols + / - = x²
Match body to "left" "right" words
Reading clocks
Fingers
Grammar
Spatial arrangement of language
III Recognizing letters and groups .ch th ing
Vision of Alphabet
Face
Lips
Tongue
Face
Lips
Inhibitions, "No"
III Recognize Word sounds
Spelling Phonics Reading
II Perceive Letter shapes d b p q
I (from eye) Lines, Angles o l - /
What NOT to SAY
Worrying (talking to yourself about what not to do)
I (from ear) Frequencies
II Phonemes
Sounds of Language
Matching vision of letters with sounds of words
Emotional memory
Language memory, stories
Emotions and language
Memory
Face Names
Occipital Lobe (vision)
Muscle Coordination
Speed of Repetitive action
Balance
Cerebellum
Left Brain
© 2001 Stephen Holland www.hiddentalents.org
Temporal Lobe (memory)


Intel’s $16.7B Acquisition of Altera



June 1, 2015

Intel to Acquire Altera

Enables New Classes of Products in High-Growth Data Center and Internet of Things Market Segments

Combination Harnesses the Power of Moore's Law to Accelerate Altera's Existing Businesses

Expected to be Accretive to Non-GAAP EPS and Free Cash Flow in First Year After Close

SANTA CLARA, Calif. & SAN JOSE, Calif.--(BUSINESS WIRE)-- Intel Corporation (NASDAQ: INTC) and Altera Corporation (NASDAQ: ALTR) today announced a definitive agreement under which Intel would acquire Altera for $54 per share in an all-cash transaction valued at approximately $16.7 billion.

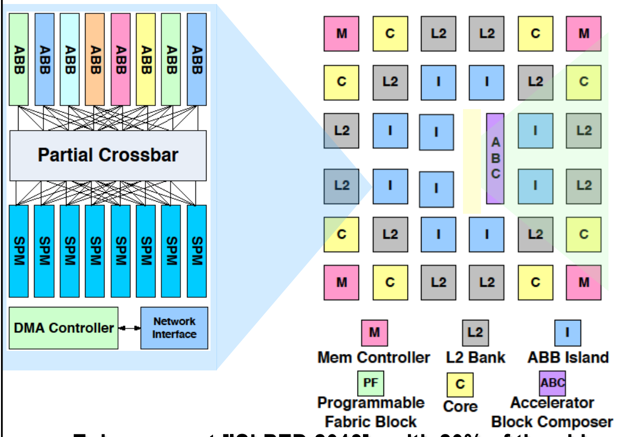
- Intel CEO Brian Krzanich noted, “The acquisition will couple Intel’s leading-edge products and manufacturing process with Altera’s leading field-programmable gate array (or FPGA) technology.” He further stated, “The combination is expected to enable new classes of products that meet customer needs in the data center and Internet of Things market segments.”

FALCON CONFIDENTIAL 7

Levels of Customization

- Single-chip level
  - Require new processor designs, e.g. using composable accelerators [ISLPED’ 12, DAC’14]
- Server node level
  - Host CPU + FPGA via PCI-e or QPI connections
- Data center level
  - Clusters of heterogeneous computing nodes

Composable Accelerators with Programmable Fabrics *[ISLPED'2013]*



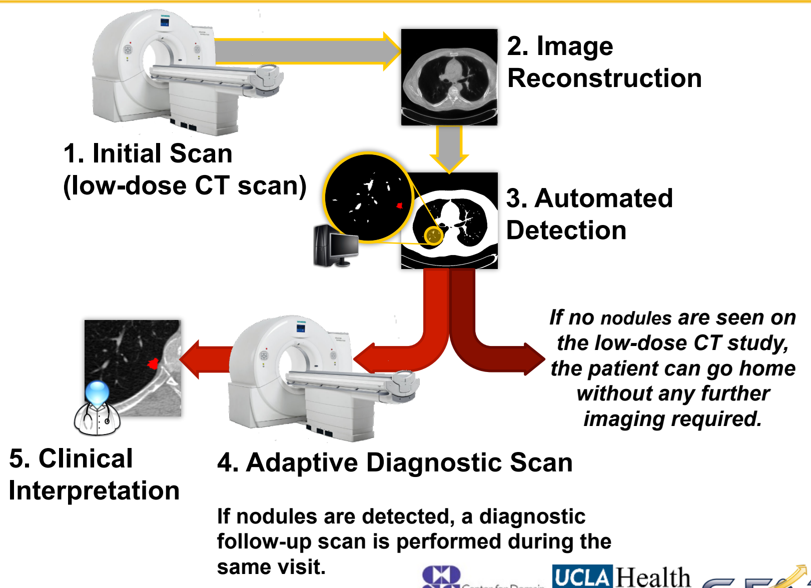
Dynamic Resource Allocation of ABBs

- **Enhancement [ISLPED 2013]: with 20% of the chip area dedicated to programmable fabric, we can achieve more:**
  - **Flexibility:** An average 8.2x (up to 146x) speedup in other domains, such as commercial, vision and navigation
  - **Longevity:** 22x speedup on a new application within the medical imaging domain

Levels of Customization

- **Single-chip level**
  - Require new processor designs, e.g. using composable accelerators [ISLPED' 12, DAC'14]
- **Server node level**
  - Host CPU + FPGA via PCI-e or QPI connections
- **Data center level**
  - Clusters of heterogeneous computing nodes

A Success Application: Low-Dose Adaptive CT Scan


1. Initial Scan
(low-dose CT scan)
2. Image Reconstruction
3. Automated Detection
If no nodules are seen on the low-dose CT study, the patient can go home without any further imaging required.
4. Adaptive Diagnostic Scan
If nodules are detected, a diagnostic follow-up scan is performed during the same visit.
5. Clinical Interpretation

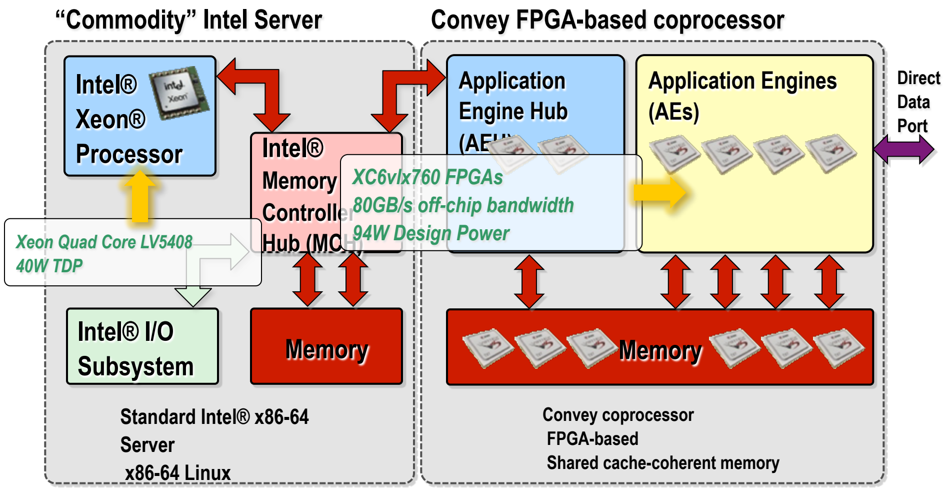
5 Years of Accelerating Medical Image Processing

	2010	2013	2015 (Today)
CT image reconstruction	18 hours Single thread CPU	20 minutes FPGA acceleration on Convey	6 minutes 4 Virtex-6 FPGAs on Convey w/data reuse
Denoising	5 minutes Single thread CPU	15 seconds NVidia GPU	3 seconds Core i7 Haswell, OpenMP, stencils
Registration	10 minutes Single thread CPU	2 minutes NVidia GPU	30 seconds Core i7 Haswell, OpenMP, stencils
Segmentation	20 minutes Single thread CPU	4 minutes Multithread CPU	1 minute Core i7 Haswell, OpenMP, stencils
Analysis	45 minutes Single thread CPU	18 minutes Multithread CPU	5 minutes* Core i7 Haswell, OpenMP

* New detection method w/improved accuracy

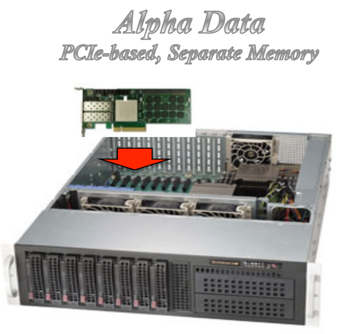
Workstation CPU, GPU, FPGA, CPU

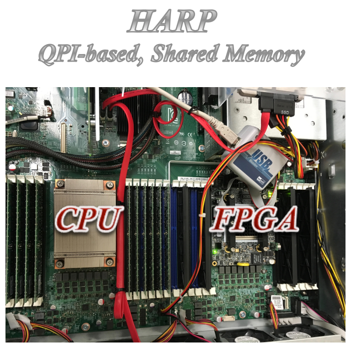
Example of CDSC Heterogeneous Computing Server


"Commodity" Intel Server
Convey FPGA-based coprocessor
Intel® Xeon® Processor
Intel® Memory Controller Hub (MCH)
Application Engine Hub (AEH)
Application Engines (AEs)
Direct Data Port
XC6vlx760 FPGAs
80GB/s off-chip bandwidth
94W Design Power
Xeon Quad Core LV5408
40W TDP
Intel® I/O Subsystem
Memory
Memory
Standard Intel® x86-64 Server
x86-64 Linux
Convey coprocessor
FPGA-based
Shared cache-coherent memory

13

More CPU-FPGA Platforms





14

14

Levels of Customization

- Single-chip level
  - Require new processor designs, e.g. using composable accelerators [ISLPED' 12, DAC'14]
- Server node level
  - Host CPU + FPGA via PCI-e or QPI connections [DAC'16]
- Data center level
  - Clusters of heterogeneous computing nodes [DAC'16]
  - How about programming at data center level? [HotCloud'16]

15

Data Center Energy Consumption is a Big Deal

In **2013**, U.S. data centers consumed an estimated **91 billion kilowatt-hours** of electricity, projected to increase to roughly **140 billion kilowatt-hours** annually by **2020**

- **50 large power plants** (500-megawatt coal-fired)
- **$13 billion annually**
- **100 million metric tons of carbon pollution per year**.

https://www.nrdc.org/resources/americas-data-centers-consuming-and-wasting-growing-amounts-energy)

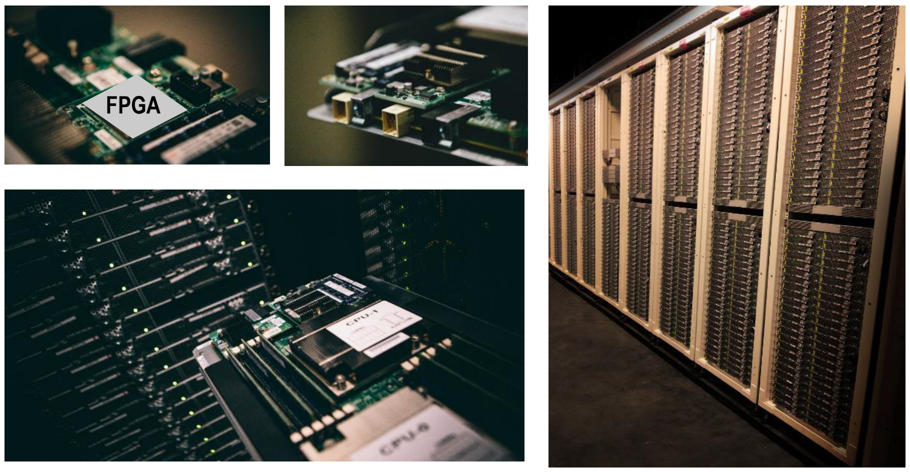
16

Extensive Efforts on Improving Datacenter Energy Efficiency

- Understand the scale-out workloads
  - ISCA'10, ASPLOS'12
  - Mismatch between workloads and processor designs;
  - Modern processors are over-provisioning
- Trade-off of big-core vs. small-core
  - ISCA'10: Web-search on small-core with better energy-efficiency
  - Baidu taps Mavell for ARM storage server SoC

17

Datacenter Level Integration at Microsoft



A. Putnam, "*A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services*", ISCA'2014

18

Focus of Our Study

- Evaluation of different integration options of heterogeneous technologies in datacenters
- Efficient programming support for heterogeneous datacenters

Small-core on Compute-intensive Workloads

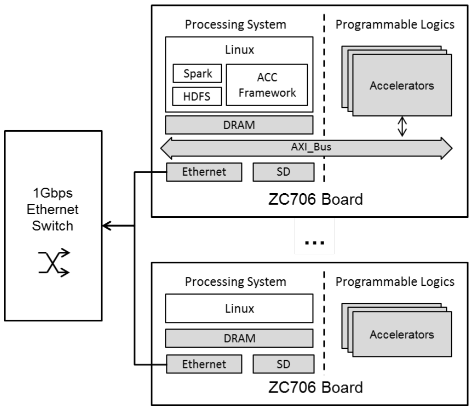
- Data set
  - MNIST 700K Samples
  - 784 Features, 10 Labels
- Benchmarks (MLLib)
  - LR: logistic regression
  - KM: k-mean clustering
- Results
  - Normalized to reference Xeon performance
- Baselines
  - Xeon: Intel E5 2620
    12 Core CPU 2.40GHz
  - Atom: Intel D2500 1.8GHz
  - ARM: A9 in Zynq 800MHz
- Power consumption (averaged)
  - Xeon: 175W/node
  - Atom: 30W/node
  - ARM: 10W/node (embedded)

Small Cores Alone Are Not Efficient!

Small Core + ACC: FARM

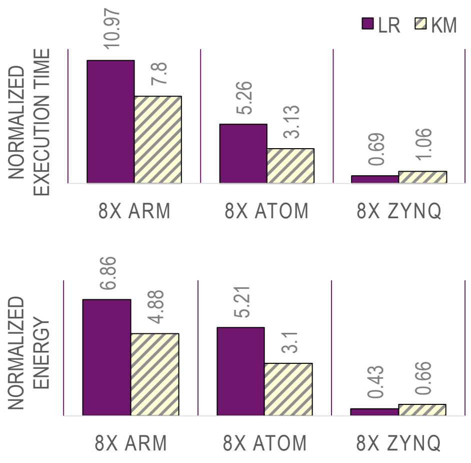
◆ Boost Small-core Performance with FPGA

- 8 Xilinx ZC706 boards
- 24-port Ethernet switch
- ~100W power



Small-core with FPGA Performance

- **Setup**
  - Data set
    - MNIST 700K Samples
    - 784 Features, 10 Labels
  - Power consumption (averaged)
    - Atom: 30W/node
    - ARM: 10W/node
- **Results**
  - Normalized to reference Xeon performance



23

Small Cores + FPGAs Are More Interesting!

24

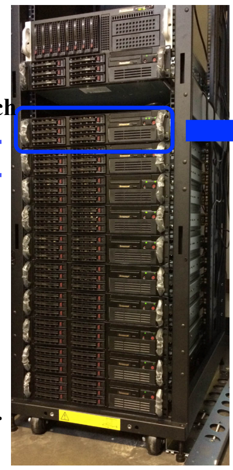
Inefficiencies in Small-core

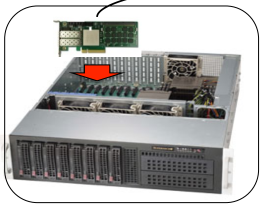
- Slower core and memory clock
  - Task scheduling is slow
  - JVM-to-FPGA data transfer is slow
- Limited DRAM size and Ethernet bandwidth
  - Slow data shuffling between nodes
- Another option: Big-core + FPGA

25

Big-Core + ACC: CDSC FPGA-Enabled Cluster

- A 24-node cluster with FPGA-based accelerators
  - Run on top of Spark and Hadoop (HDFS)





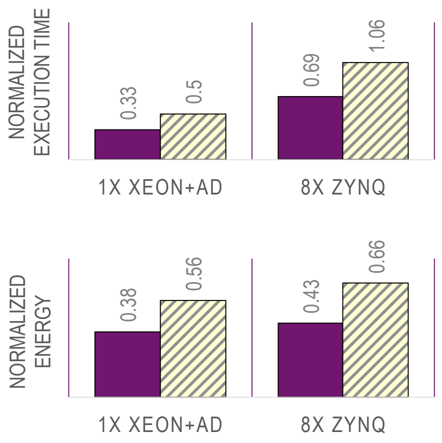
Alpha Data board:
1. Virtex-7 FPGA
2. 16GB on-board RAM

Each node:
1. Two Xeon processors
2. One FPGA PCIe card (Alpha Data)
3. 64 GB RAM
4. 10GBE NIC

26

Experimental Results

- Experimental setup
  - Data set
    - MNIST 700K Samples
    - 784 Features, 10 Labels
- Results
  - Normalized to reference Xeon performance



27

Overall Evaluation Results

- Based on two machine learning workloads
  - Normalized performance (speedup), and energy efficiency (performance/W) relative to big-core solutions

	Performance	Energy-Efficiency
Big-Core+FPGA	Best | 2.5	Best | 2.6
Small-Core+FPGA	Better | 1.2	Best | 1.9
Big-Core	*Good | 1.0*	*Good | 1.0*
Small-Core	Bad | 0.25	Bad | 0.24

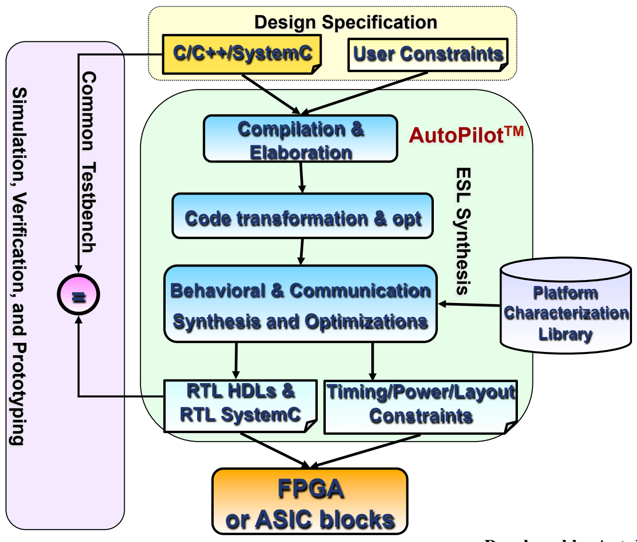
28

How to Program Such "Beasts"?

-- "Write Once, Accelerate Anywhere"

29

C/C++ Based Synthesis for Accelerator Design
xPilot (UCLA 2016) -> AutoPilot (AutoESL) -> Vivado HLS (Xilinx 2011-)


Design Specification
C/C++/SystemC
User Constraints
Simulation, Verification, and Prototyping
Common Testbench
=
Compilation & Elaboration
AutoPilot™
ESL Synthesis
Code transformation & opt
Behavioral & Communication Synthesis and Optimizations
Platform Characterization Library
RTL HDLs & RTL SystemC
Timing/Power/Layout Constraints
FPGA or ASIC blocks

- Platform-based C to RTL synthesis
- Synthesize pure ANSI-C and C++, GCC-compatible compilation flow
- Full support of IEEE-754 floating point data types & operations
- Efficiently handle bit-accurate fixed-point arithmetic
- SDC-based scheduling
- Automatic memory partitioning
- …

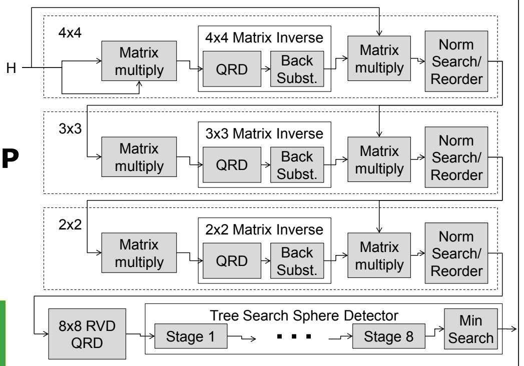
QoR matches or exceeds manual RTL for many designs

Developed by AutoESL, acquired by Xilinx in Jan. 2011

30

AutoPilot Results: Sphere Decoder (from Xilinx)

- **Wireless MIMO Sphere Decoder**
  - **~4000 lines of C code**
  - **Xilinx Virtex-5 at 225MHz**
- **Compared to optimized IP**
  - **11-31% better resource usage**



Metric	RTL Expert	AutoPilot Expert	Diff (%)
LUTs	32,708	29,060	-11%
Registers	44,885	31,000	-31%
DSP48s	225	201	-11%
BRAMs	128	99	-26%

TCAD April 2011 (keynote paper) "High-Level Synthesis for FPGAs: From Prototyping to Deployment"

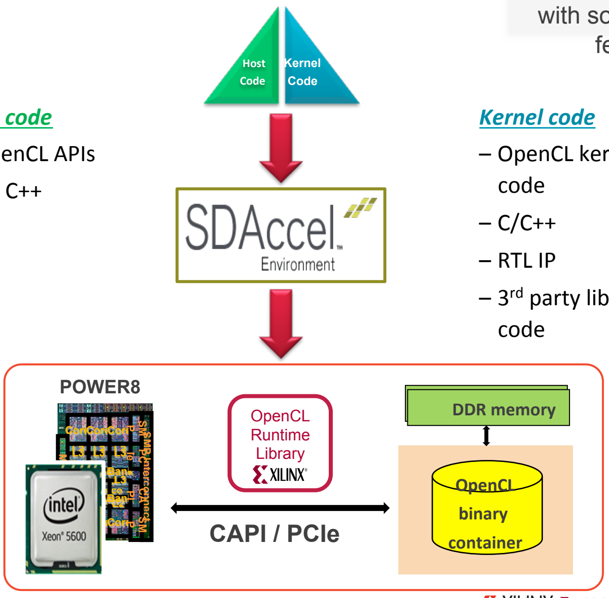
31

SDAccel Development Environment

SDAccel supports OpenCL 1.0 Embedded Profile with some 1.2 / 2.0 features

Host code
- OpenCL APIs
- C / C++

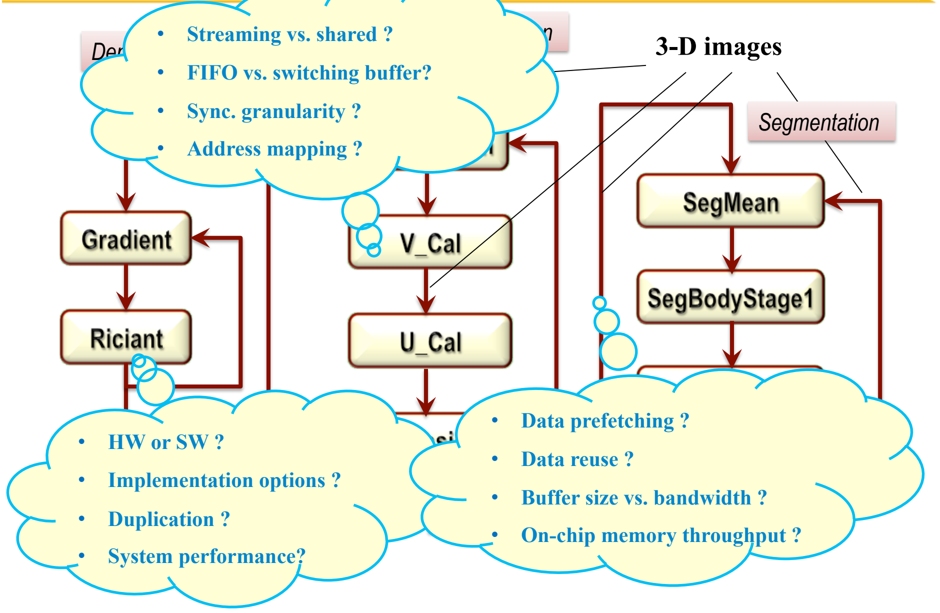
Kernel code
- OpenCL kernel code
- C/C++
- RTL IP
- 3rd party library code



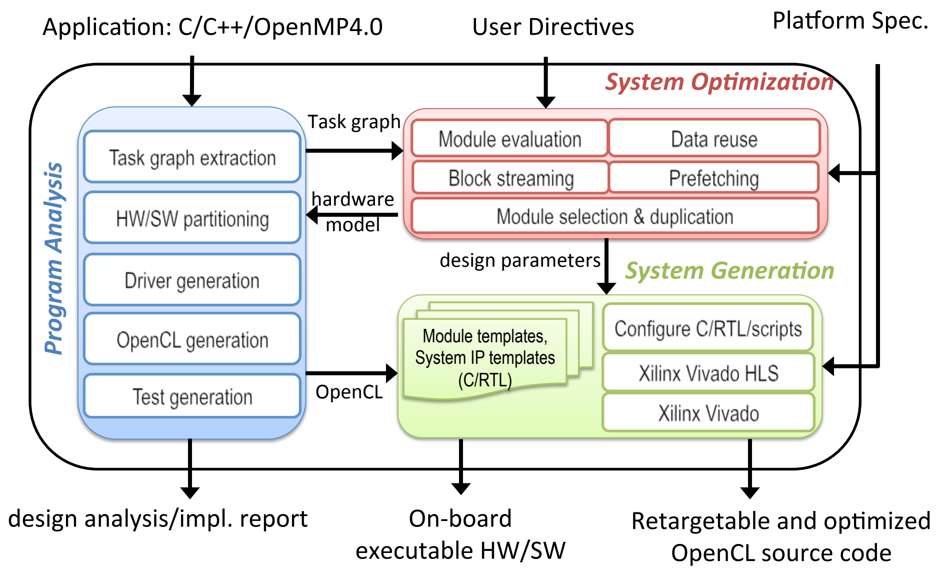
Page 32

XILINX ALL PROGRAMMABLE.

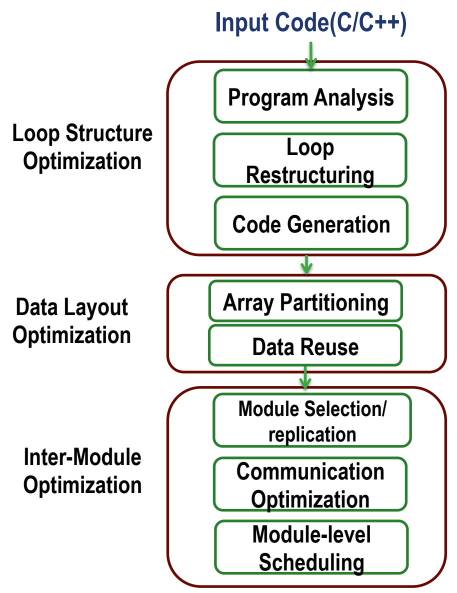
Design Complexity Can Still be High – Example: Medical Image Processing Pipeline



CMOST: Fully Automated Compilation and Mapping Flow [DAC 2015]



Recent Research - Optimizations Beyond HLS



Polyhedral-Based Data Reuse Optimization for Configurable Computing FPGA'13 Best Paper Award

Improving Polyhedral Code Generation for High-Level Synthesis CODES-ISSS'14 Best Paper Award

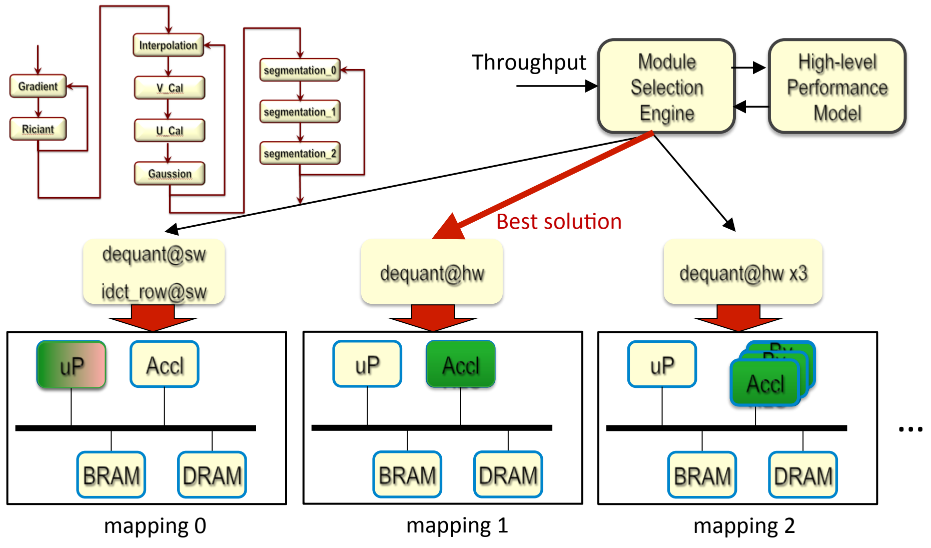
Theory and Algorithm for Generalized Memory Partitioning in High-Level Synthesis, FPGA'14

An optimal microarchitecture for stencil computation acceleration based on non-uniform partitioning of data reuse buffers (DAC'14)

Combining Computation with Communication Optimization in System Synthesis for Streaming Applications, FPGA'14

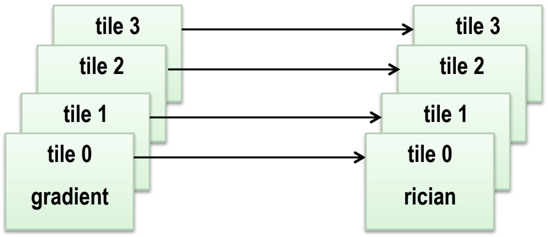
35

Example: Throughput-Driven Task Scheduling and Mapping [FPGA'2014]



Motivation

Tile size: 32x32
Image: 64x64, 4 tiles



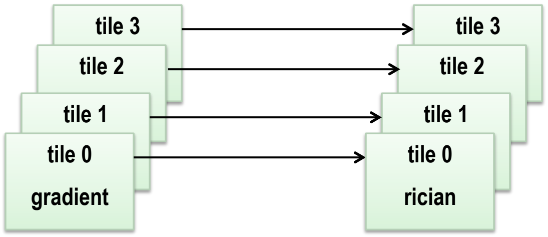
- Which implementation to use for each module?
  - Memory partitioned v.s. non-memory-partitioned

	BRAM	DSP	FF	LUT
non-partitioned gradient	128	21	2511	2125
partitioned gradient	176	56	7147	7262
partitioned rician	128	22	4692	3991
non-partitioned rician	176	88	14475	15537

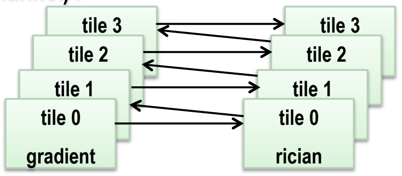
37

Motivation

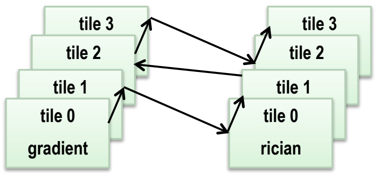
Tile size: 32x32
Image: 64x64, 4 tiles



- How many number of replicas?
- Scheduling and Communication cost (number of tiles in the communication channel)?



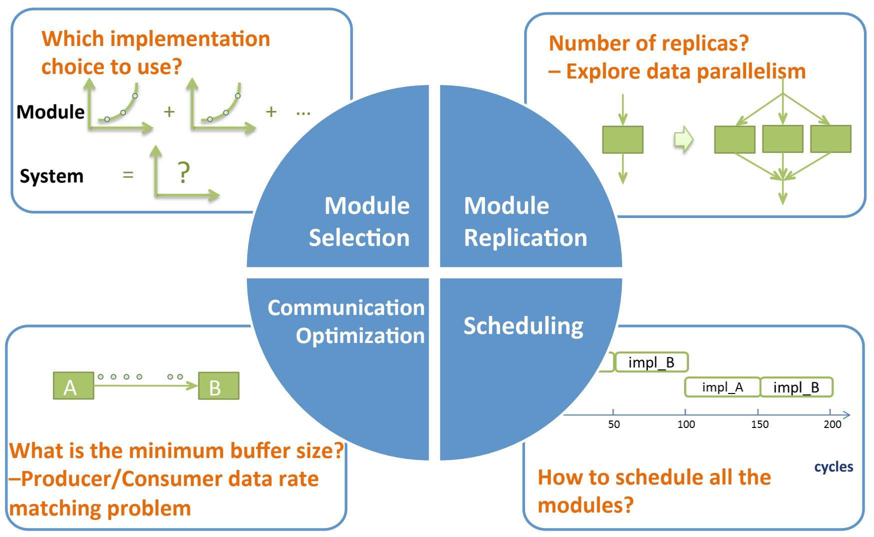
scheduling 0 →1 tile



scheduling 0 →2 tiles

38

A Rich Design Space: System-Level Synthesis with HLS for Streaming Applications

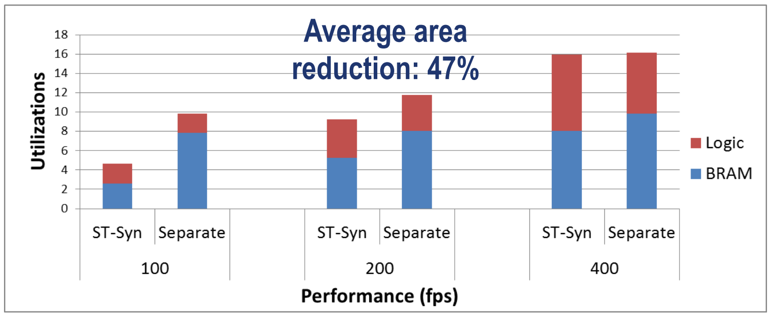


Which implementation choice to use?
Module
+
+ …
System
=
?
Number of replicas?
– Explore data parallelism
Module Selection
Module Replication
Communication Optimization
Scheduling
A
B
What is the minimum buffer size?
–Producer/Consumer data rate matching problem
impl_B
impl_A
impl_B
50
100
150
200
cycles
How to schedule all the modules?

39

Experiments on Example Denoise

- Our methodology: ST-Syn
  - computation & communication co-optimization
- Separate:
  - separate computation opt. + communication opt.
- → Communication and computation should be considered in a unified framework



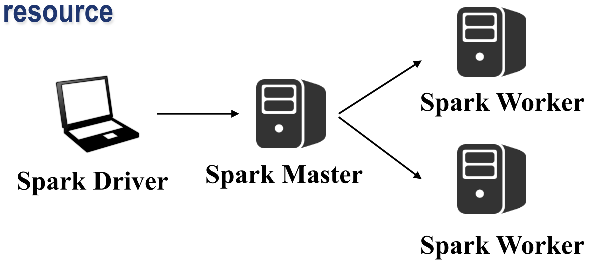
40

More is Needed for Data Center Level Deployment

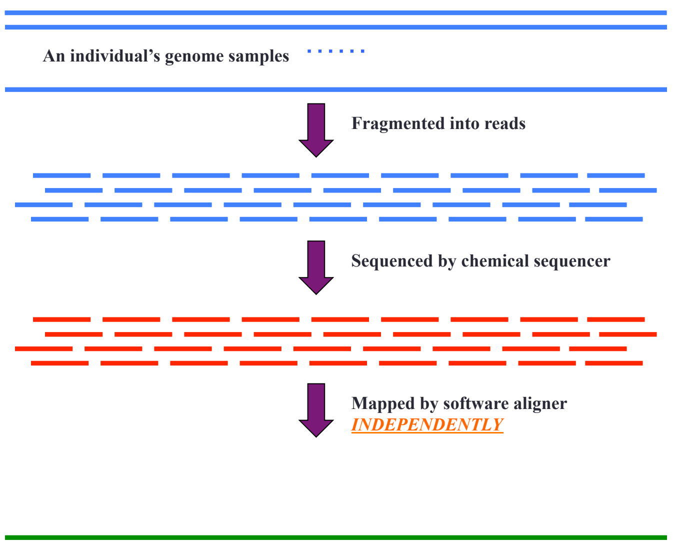
Scalable Big-Data Programming

- Simplified programming models
  - MapReduce, Dataflow
- User-transparent Runtime
  - Distributed computing
  - Scheduling and resource management
  - Fault-tolerance

```
val points = sc.textfile().cache()
for (i <- 1 to ITERATIONS) {
  val gradient = points.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x)))
    - 1) * p.y * p.x
    ).reduce(_ + _)
  w -= gradient
}
```



Next-Generation DNA Sequencing


An individual's genome samples
Fragmented into reads
Sequenced by chemical sequencer
Mapped by software aligner
INDEPENDENTLY

43

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

44

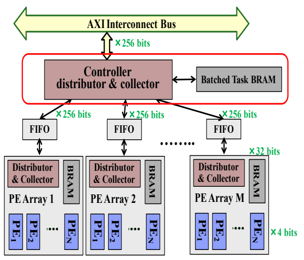
Straightforward Integration: 1+1 < 0.001

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

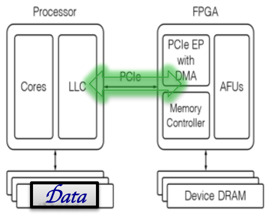
On FPGA
One DP
⇒ 25 ms data transfer
⇒ 1.6 reads/s

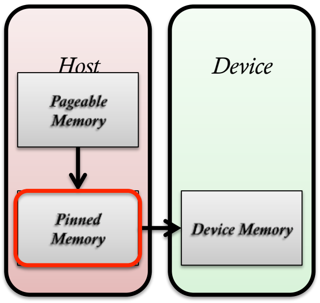
While *JNI* serves as a standard approach to connect JVMs with FPGAs,
a straightforward integration through JNI degrades the performance by ***1000x***.

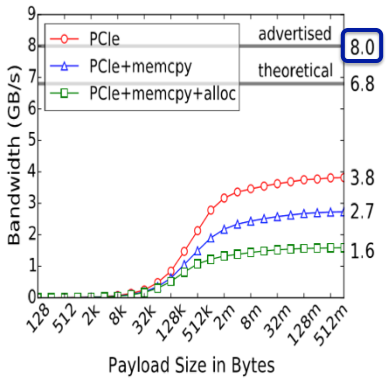
45

What happened in a CPU-FPGA communication instance?

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







46

Why communication matters?

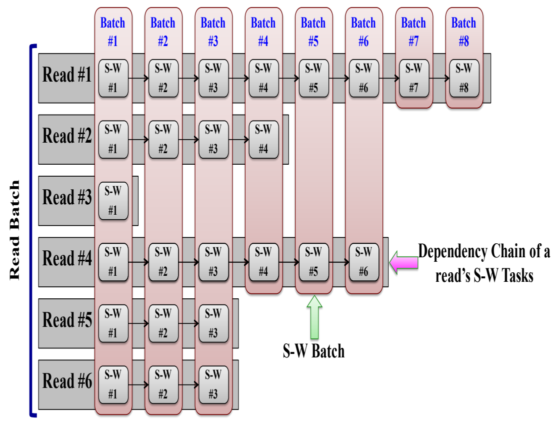
- Each map function is likely to process only a small volume of data with a small amount of execution time
  - One read is only 101 ASCII characters
  - One line of a text file
  - One record of a NoSQL table
  - …
- Communication overhead can be amortized by batch processing

```
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))
```

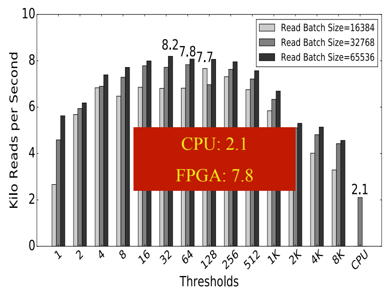
```
def map_func(input:Array[U]):Array[V] = {
  // Array[U] => … => Array[V]
  t1:Array[P] = cnv1_batch(input)
  t2:Array[Q] = cnv2_batch(t1)
  t3:Array[V] = cnv3_batch(t2)
  t3
}
rdd_out = rdd_in.map(ele=>map_func(ele))
```

47

Let's first do batch processing manually

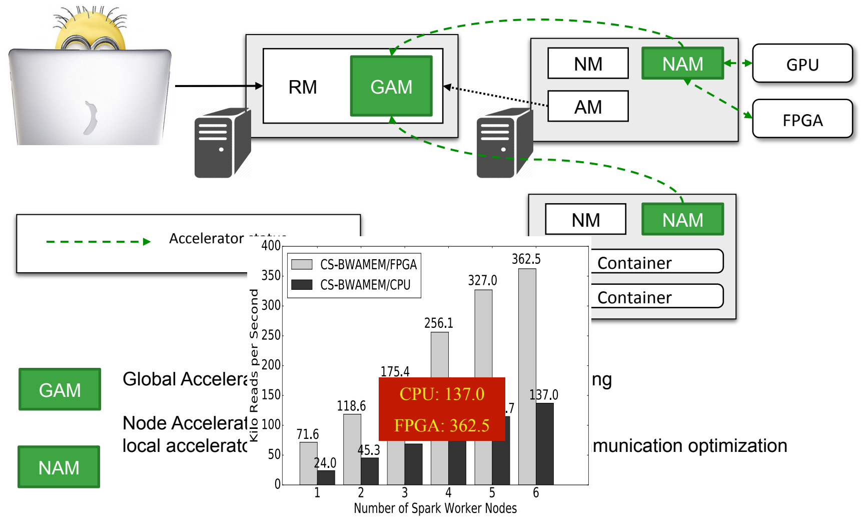


Dependency/Irregularity-Aware Batch Processing



48

Accelerator-as-a-Service

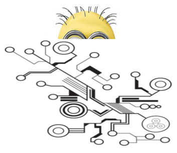


Source: https://spark-summit.org/2016/events/deploying-accelerators-at-datacenter-scale-using-spark/

49

Blaze Runtime System

- A system providing Accelerator-as-a-Service
  - Provide a better programming model:
    - APIs for accelerator developers
      - Easier to integrate into big-data workload, e.g. Spark and Hadoop
    - APIs for big-data application developers
      - Requires no knowledge about accelerators
  - Provide an accelerator management runtime
    - Supports FPGAs and GPUs

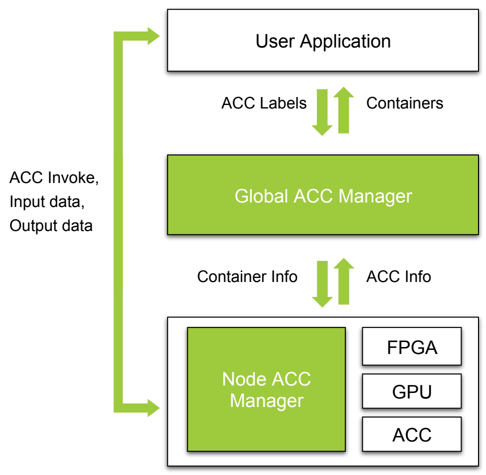


Source: https://spark-summit.org/2016/events/deploying-accelerators-at-datacenter-scale-using-spark/

50

Runtime Flow

- Accelerator Registration
  - Register accelerator service to corresponding nodes
- Job Accelerator Request
  - Use acc_id as label
  - GAM allocates containers to corresponding nodes
- Job execution
  - Adopts several optimization techniques, e.g. Double-buffering, caching

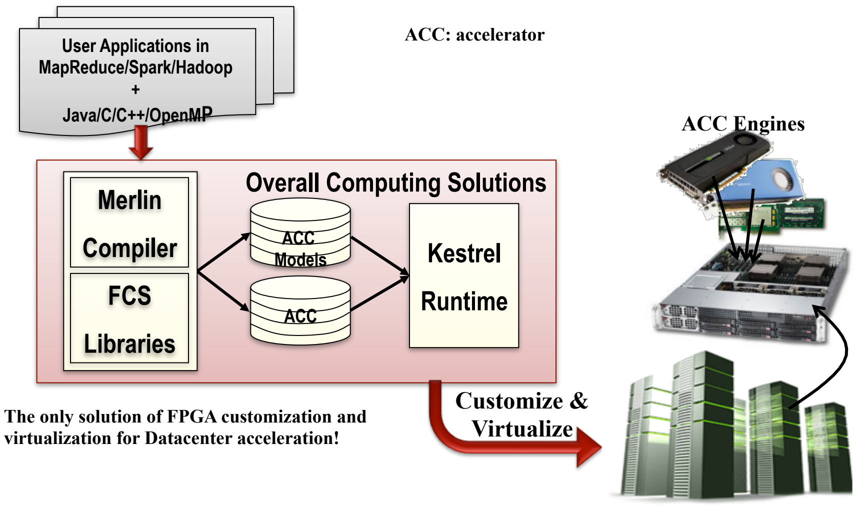


Source: https://spark-summit.org/2016/events/deploying-accelerators-at-datacenter-scale-using-spark/

Falcon Computing Solutions, Inc.

http://www.falcon-computing.com

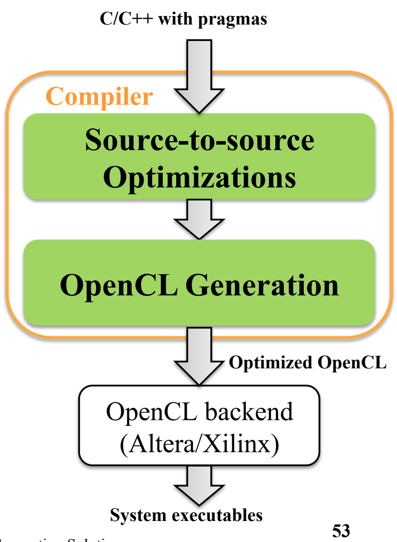
ACC: accelerator


User Applications in MapReduce/Spark/Hadoop + Java/C/C++/OpenMP
Overall Computing Solutions
Merlin Compiler
FCS Libraries
ACC Models
ACC
Kestrel Runtime
ACC Engines
Customize & Virtualize

The only solution of FPGA customization and virtualization for Datacenter acceleration!

Merlin Compiler

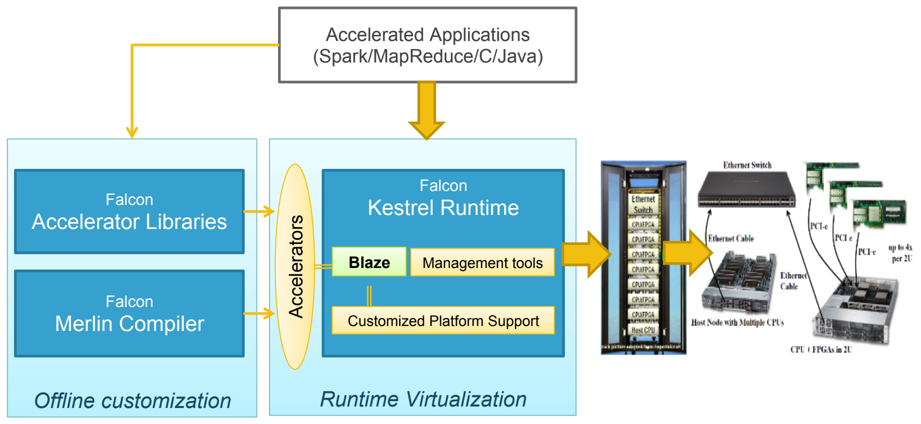
- C-based design flow
- OpenMP-like high-level programming model
- Automatic optimizations for productivity and QoR
- Same input for multi-vendors and multi-platforms



Sample Compilation Results

Design	Merlin Compiler	Initial OpenCL	Manual Optimized OpenCL
Blackschole	0.34ms	11ms	NA
Denoise	0.08s	3.8s	NA
LogisticRegr	94ms	3.7s	94ms
MatMult	0.8ms	1.9ms	0.8ms
NAMD	26ms	51ms	26ms
Normal	4ms	52ms	10ms
TwoNN	1.23s	1.70s	NA
Average	**1x**	**21x**	**1.3x**

Kestrel Runtime And Blaze



55

Concluding Remarks

- New era of computing
  - Accelerator-centric computing
  - Need efficient support for customization and specialization
- Customization at all levels
  - Chip-level
  - Server node level
  - Data center level
- Data center level customization holds great promise
  - That's where workload aggregates
- Software is the key
  - Programming models
    - Hadoop/MapReduce or SPARK (+ C/C++), OpenMP, OpenCL,, …
  - Compilation support
  - Runtime management

56

Acknowledgements – CDSC and C-FAR

- Center for Domain-Specific Computing (CDSC) under the NSF Expeditions in Computing Program and C-FAR Center under the STARnet Program
- CDSC faculty:

Aberle (UCLA)

Baraniuk (Rice)

Bui (UCLA)

Chang (UCLA)

Cheng (UCSB)

Cong (Director) (UCLA)

Palsberg (UCLA)

Potkonjak (UCLA)

Reinman (UCLA)

Sadayappan (Ohio-State)

Sarkar (Associate Dir) (Rice)

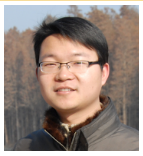
Vese (UCLA)

57

Postdocs, Graduate Students, and Collaborators

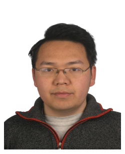
Prof. Deming Chen (UIUC/ADSC)

Yuting Chen (UCLA)


Zhenman Fang (UCLA)

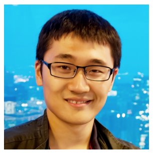
Hui Huang (UCLA)

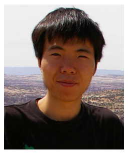
Muhuan Huang (UCLA)


Dr. Peng Li (UCLA)

Prof. Louis-Noël Pouchet (UCLA)

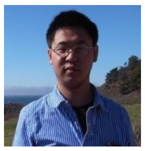
Yuxin Wang (PKU)


Di Wu (UCLA)


Bingjun Xiao (UCLA)

Hao Yu (UCLA)

Dr. Peng Zhang (UCLA)


Yi Zou (UCLA)

Wei Zuo (UIUC)

58