Scaling Big Data Analytics with Moore’s Law

Kunle Olukotun
EE and CS
Data Trends

- Increasing volume, variety and complexity of data
- Challenge: enable data-driven discovery
  - Deliver the capability to mine, search and analyze this data in near real time
Microprocessor Trends

Moore's Law
End of sequential performance

Transistors (Thousands)
Single-Thread Performance (SpecINT)
Frequency (MHz)
Typical Power (Watts)
Number of Cores
Power Wall

Data collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, C. Batten
Heterogeneous Computing Platforms

Parallelism and specialization

- **Graphics Processing Unit (GPU)**: > 8 TFLOPS, SIMD
- **Accelerators**: > 9 TFLOPS
- **Programmable Logic**: > 1 TB DRAM

- **Cluster**: 1000s of nodes
- **Multicore Multi-socket NUMA**: 10s of cores, SIMD
- **10s of cores, SIMD**: > 1 TB DRAM
Specialized Parallel Programming

PPP

CUDA
OpenCL

Multicore CPU
Muti-socket

Threads
OpenMP

Verilog
VHDL

Graphics
Processing
Unit (GPU)

Programmable
Logic

Custom computing

MPI: Message Passing Interface

Cluster

MPI
Map Reduce
Specialized Programmers ⇒ Scarce
Data Analytics Programming Challenge

Data Analytics Application

- Data ETL
- Data Query
- Graph Analysis
- Predictive Analytics

Pthreads
OpenMP

Multicore

CUDA
OpenCL

GPU

MPI
Map Reduce

Cluster

Verilog
VHDL

FPGA
Data Analytics Programming Challenge

Data Analytics Application

- Data ETL
- Data Query
- Graph Analysis
- Predictive Analytics

High-Performance Domain Specific Languages

Multicore

- Pthreads
- OpenMP

GPU

- CUDA
- OpenCL

Cluster

- MPI
- Map Reduce

FPGA

- Verilog
- VHDL
Domain Specific Languages

- Domain Specific Languages (DSLs)
  - Programming language with restricted expressiveness for a particular domain
  - High-level, usually declarative, and deterministic

![OpenGL](image1)
![MATLAB](image2)
![Structured Query Language](image3)
![RAILS](image4)
![TEX](image5)
High Performance DSLs for Data Analytics

Applications
- Data Transformation
- Graph Analysis
- Prediction Recommendation

HP DSLs
- Data Extraction: OptiWrangle
- Query Proc.: OptiQL
- Graph Alg.: OptiGraph
- Machine Learning: OptiML

DSL Compilers

Heterogeneous Hardware
- Multicore
- GPU
- FPGA
- Cluster
Scaling the HP DSL Approach

- Many potential DSLs

- How do we quickly create high-performance implementations for DSLs we care about?

- Enable expert programmers to easily create new DSLs
  - Make optimization knowledge reusable
  - Simplify the compiler generation process

- A few DSL developers enable many more DSL users
  - Leave expert programming to experts!
Delite: DSL Infrastructure

Applications
- Data Transformation
- Graph Analysis
- Prediction Recommendation

HP DSLs
- Data Extraction OptiWrangle
- Query Proc. OptiQL
- Graph Alg. OptiGraph
- Machine Learning OptiML

Delite DSL Framework
- DSL Compiler
- DSL Compiler
- DSL Compiler
- DSL Compiler

Heterogeneous Hardware
- Multicore
- GPU
- FPGA
- Cluster
Delite: A Framework for High Performance DSLs

- A compiler tool-chain for high performance embedded DSLs
  - Libraries on steroids (generative programming)

- Built on top of Lightweight Modular Staging (LMS) to build an intermediate representation (IR) from Scala application code

- Provides extensible reusable components
  - Parallel patterns for structured computation
  - Delite structs for structured data
  - Transformers for domain-specific optimizations

- Delite optimizes DSL code and generates target code
  - Scala, C++, CUDA, OpenCL, and clusters
Parallel Patterns

Most data analytic computations can be expressed as functional parallel patterns on collections (e.g. sets, arrays, tables, n-d matrices)

Nested parallel patterns

Map, Zip, Filter, FlatMap, Reduce, GroupBy, Join, Sort, ...

**Key elements**
- DSLs embedded in Scala
- IR created using type-directed staging
- Domain specific optimization
- General parallelism and locality optimizations
- Optimized mapping to HW targets
OptiML: Overview

- Provides a familiar (MATLAB-like) language and API for writing ML applications
  - Ex. `val c = a * b` (a, b are Matrix[Double])

- Implicitly parallel data structures
  - Base types
    - Vector[T], Matrix[T], Graph[V,E], Stream[T]
  - Subtypes
    - TrainingSet, IndexVector, Image, ...

- Implicitly parallel control structures
  - `sum{...}, (0::end) {...}, gradient { ... }, untilconverged { ... }
  - Allow anonymous functions with restricted semantics to be passed as arguments of the control structures
K-means Clustering in OptiML

until converged(kMeans, tol) {
    val clusters = samples.groupRowsBy {
        sample =>
            kMeans.mapRows(mean => dist(sample, mean)).minIndex
    }
    val newKmeans = clusters.map(c => c.sum / c.length)
    newKmeans
}

• No explicit map-reduce, no key-value pairs (e.g. MR)
• No distributed data structures (e.g. Spark RDDs)
• No annotations for hardware design
• Efficient multicore and GPU execution
• Efficient cluster and NUMA implementation
• Efficient FPGA hardware
Mapping Nested Parallel Patterns to GPUs

\[ m = \text{Matrix.rand}(nR,nC) \]
\[ v = m.\text{sumCols} \]

\[ m = \text{Matrix.rand}(nR,nC) \]
\[ v = m.\text{sumRows} \]

HyoukJoong Lee et. al, “Locality-Aware Mapping of Nested Parallel Patterns on GPUs,” MICRO'14
Markov State Models (MSMs)

MSMs are a powerful means of modeling the structure and dynamics of molecular systems, like proteins.
Distributed Heterogeneous Execution

- Separate Memory Regions
  - NUMA
  - Clusters
  - FPGAs

- Partitioning Analysis
  - Multidimensional arrays
  - Decide which data structures / parallel ops to partition across abstract memory regions

- Nested Pattern Transformations
  - Optimize patterns for distributed and heterogeneous architectures
Heterogeneous Cluster Performance

4 node local cluster: 3.4 GB dataset
Multi-socket NUMA Performance
Multi-Terabyte Main Memory

TPCHQ1 | Gene | GDA | LogReg

1 – 48 threads
4 sockets
SIMD Parallelism (Intel AVX2)

- Single Instruction Multiple Data (SIMD)
  - SIMD parallelism is keeping up with Moore’s law: doubling per generation
- Precision vs. parallelism

SIMD Precision

32-bit float vector

16-bit int vector

8-bit int vector

SIMD Parallelism

8 multiplies/cycle
(vmulps instruction)

16 multiplies/cycle
(vpmaddwd instruction)

32 multiplies/cycle
(vpmaddubsw instruction)

Tradeoff between precision & parallelism
Statistical vs. Hardware Efficiency with Chris Ré

**Same statistical efficiency**

![Graph showing logistic regression using SGD with 32-bit and 8-bit precision.](image)

- 8-bit gives about 3x speed up!
- Lower precision is possible
- Good match to specialized/reconfigurable HW?

**Improved hardware efficiency**

BUCKWILD! same **statistical efficiency** with greater **hardware efficiency**
FPGAs in the Datacenter?

- FPGAs based accelerators
  - Recent commercial interest from Baidu, Microsoft, and Intel
  - Key advantage: Performance, Performance/Watt
  - Key disadvantage: lousy programming model

- Verilog and VHDL poor match for software developers
  - But, high quality designs

- High level synthesis (HLS) tools with C interface
  - Medium/low quality designs
  - Need architectural knowledge to build good accelerators
  - Not enough information in compiler IR to perform access pattern and data layout optimizations
  - Cannot synthesize complex data paths with nested parallelism
Our Approach to FPGA Design

High-level Parallel Patterns

Data Locality improved with parallel pattern tiling transformations

Nested Parallelism exploited with hierarchical pipelines and double buffers

Generate MaxJ to generate VHDL

Parallel Patterns

Pattern Transformations
  - Fusion
  - Pattern Tiling
  - Code Motion

Tiled Parallel Pattern IR

Hardware Generation
  - Memory Allocation
  - Template Selection
  - Metapipeline Analysis

MaxJ HGL

Bitstream Generation

FPGA Configuration
K-Means Hardware

1. Load kmeans

2. Metapipeline: Calculate **sum** and **count**

3. Metapipeline: Calculate **new kmeans**, store results

Similar to (and more general than) hand-written designs\(^1\)

Impact of Tiling and Metapipelining

![Bar Chart for Speedup](chart1.png)

![Bar Chart for Resource Use](chart2.png)
Scaling Big Data Analytics with Moore’s Law

Applications
- Data Transformation
- Graph Analysis
- Prediction Recommendation

HP DSLs
- Data Extraction OptiWrangle
- Query Proc. OptiQL
- Graph Alg. OptiGraph
- Machine Learning OptiML

Delite DSL Framework
- Parallel data
- Parallel patterns
- Analyses & Transformations

Heterogeneous Hardware
- Multicore
- GPU
- Cluster
- FPGA
Colaborators & Funding

Faculty
- Pat Hanrahan
- Martin Odersky (EPFL)
- Chris Ré
- Tiark Rompf (Purdue/EPFL)

PhD Students
- Chris Aberger
- Kevin Brown
- Hassan Chafi
- Zach DeVito
- Chris De Sa
- Nithin George (EPFL)
- David Koeplinger

Funding
- PPL : Oracle Labs, Nvidia, Intel, AMD, Huawei, SAP
- NSF
- DARPA

- HyoukJoong Lee
- Victoria Popic
- Raghu Prabhakar
- Aleksander Prokopec (EPFL)
- Vera Salvisberg (EPFL)
- Arvind Sujeeth
Scaling Big Data Analytics with Moore’s Law

- Increasing volume, variety and complexity of data
- Heterogeneity of modern hardware
- Serious challenges for data analytics

- Power
- Performance
- Programmability
- Portability

Modern HW (Multicore, SIMD, GPU, FPGA, NUMA)

Delite

High Performance DSLs (OptiML, OptiQL, ...)

Increasing volume, variety and complexity of data
Heterogeneity of modern hardware
Serious challenges for data analytics