High Performance Big-Data Analytics

Kunle Olukotun
Pervasive Parallelism Laboratory
Stanford University
ppl.stanford.edu

Big Data Analytics Today

Disk-to-disk map-reduce data processing

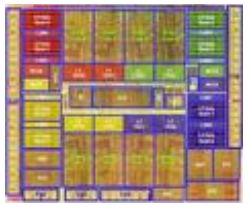
=

Next Generation Big Data Analytics: Improved Decision Making

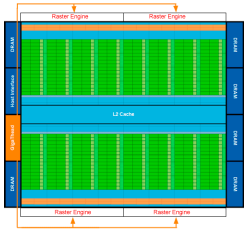
- Higher performance ⇒ faster decisions
  - Bigger data sizes ⇒ better decisions
  - Low latency big data processing ⇒ interactive decisions
  - Processing on live data streams ⇒ real time decisions
- Higher productivity ⇒ easier decisions
  - More intuitive than map-reduce with key-value pairs
  - Simple programming for complex tasks
    - Data transformation
    - Graph analysis
    - Predictive analysis using machine learning

Next Gen Big Data Analytics Must Embrace Heterogeneous Parallelism

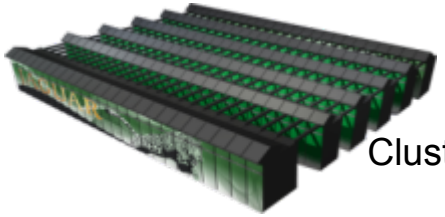
Fine grained parallelism is the only way to get high performance and performance/watt



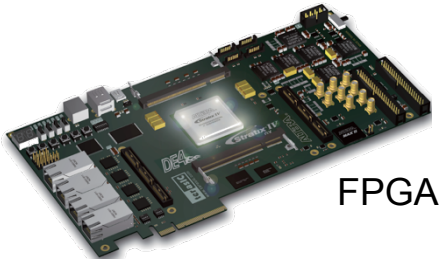
Multicore



GPU



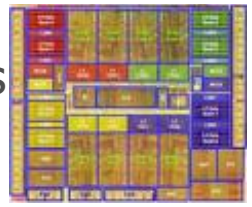
Cluster



FPGA

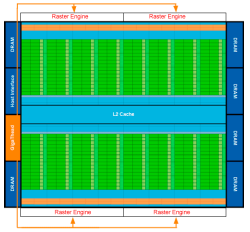
Heterogeneous Parallel Programming

Pthreads
OpenMP



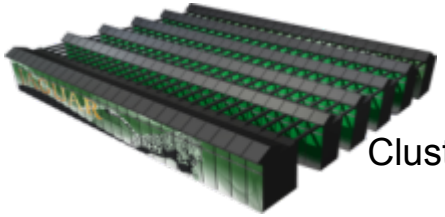
Muticore

CUDA
OpenCL



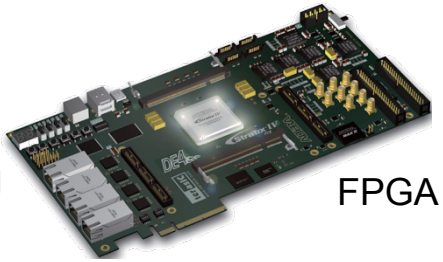
GPU

MPI
PGAS



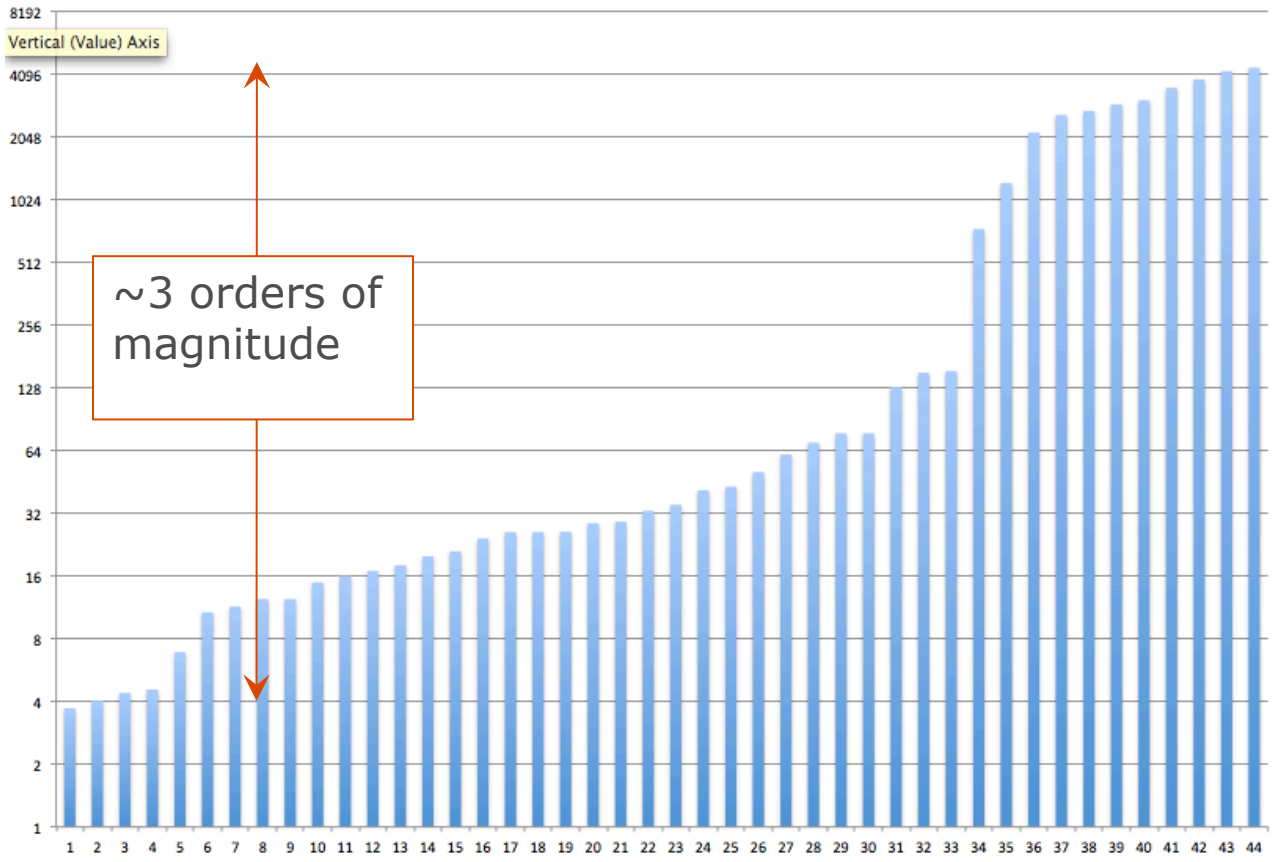
Cluster

Verilog
VHDL



FPGA

Huge Performance Variation: Image Filtering OpenMP Assignment



Optimizations:

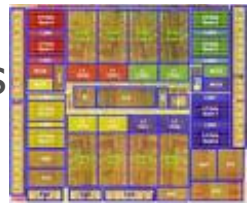
- Precomputing twiddle
- Not computing what isn't part of the filtering
- Transposing the matrix
- Using SSE

Big-Data Analytics Programming Challenge

Data Analytics Application

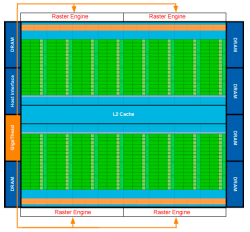
- **Data Prep**
- **Data Transform**
- **Network Analysis**
- **Prediction**

Pthreads
OpenMP



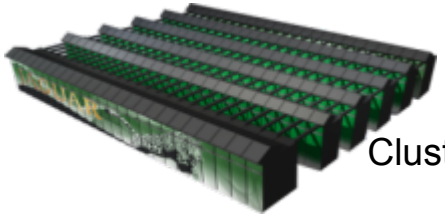
Multicore

CUDA
OpenCL



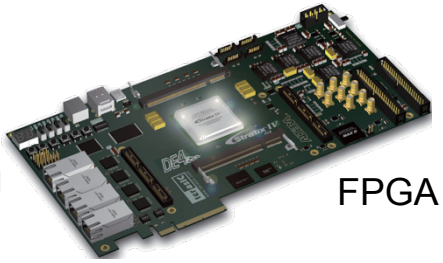
GPU

MPI
PGAS



Cluster

Verilog
VHDL



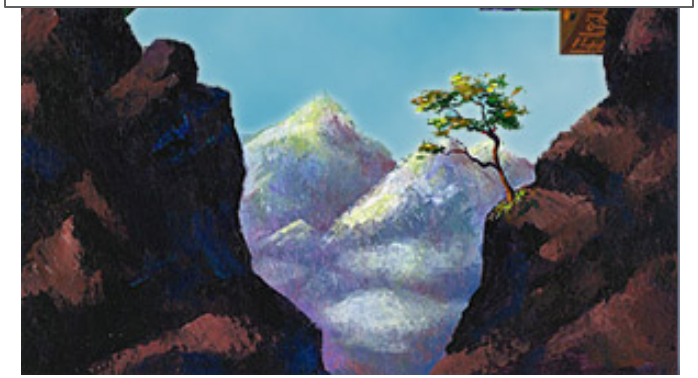
FPGA

Big-Data Analytics Programming Challenge

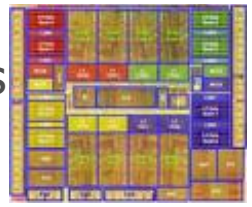
Data Analytics Application

- **Data Prep**
- **Data Transform**
- **Network Analysis**
- **Prediction**

Domain Specific Languages

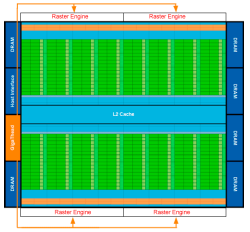


Pthreads
OpenMP



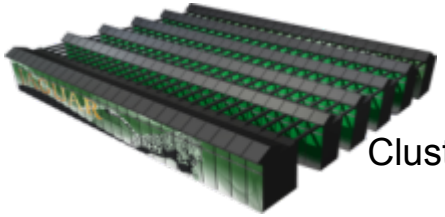
Multicore

CUDA
OpenCL



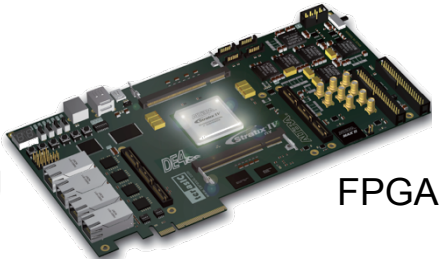
GPU

MPI
PGAS



Cluster

Verilog
VHDL



FPGA

Domain Specific Languages

- Domain Specific Languages (DSLs)
  - Definition: A language or library with restrictive expressiveness that exploits domain knowledge for productivity and efficiency
  - High-level, usually declarative, and deterministic

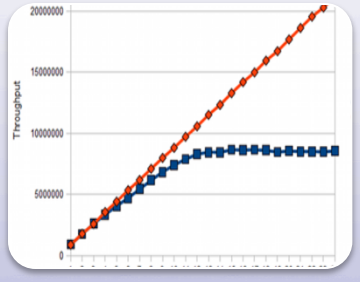
Benefits of Using DSLs for High Performance

Productivity

- Shield most programmers from the difficulty of parallel programming
- Focus on developing algorithms and applications and not on low level implementation details

Performance

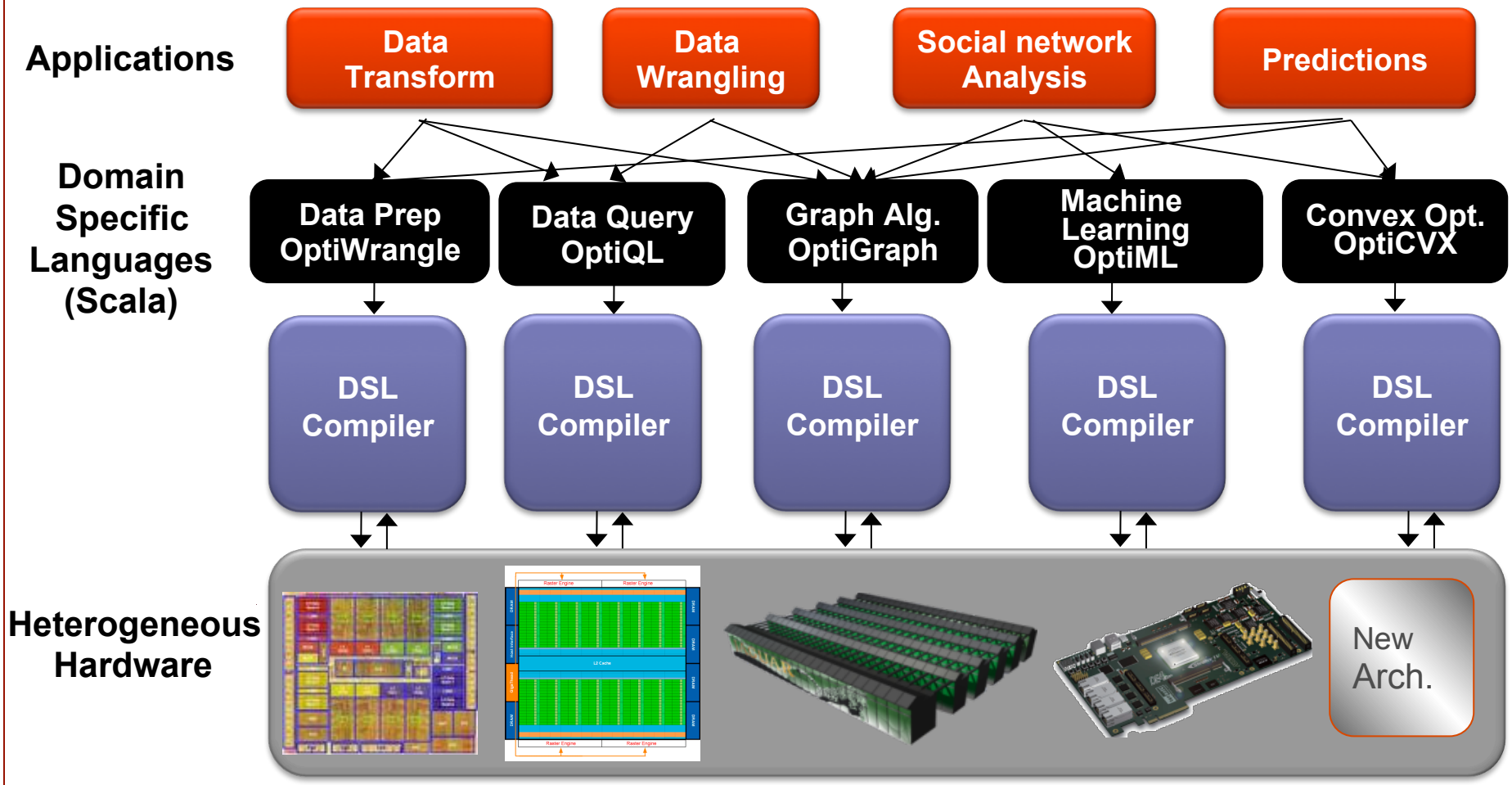
- Match high level domain abstraction to generic parallel execution patterns
- Restrict expressiveness to more easily and fully extract available parallelism
- Use domain knowledge for static/dynamic optimizations



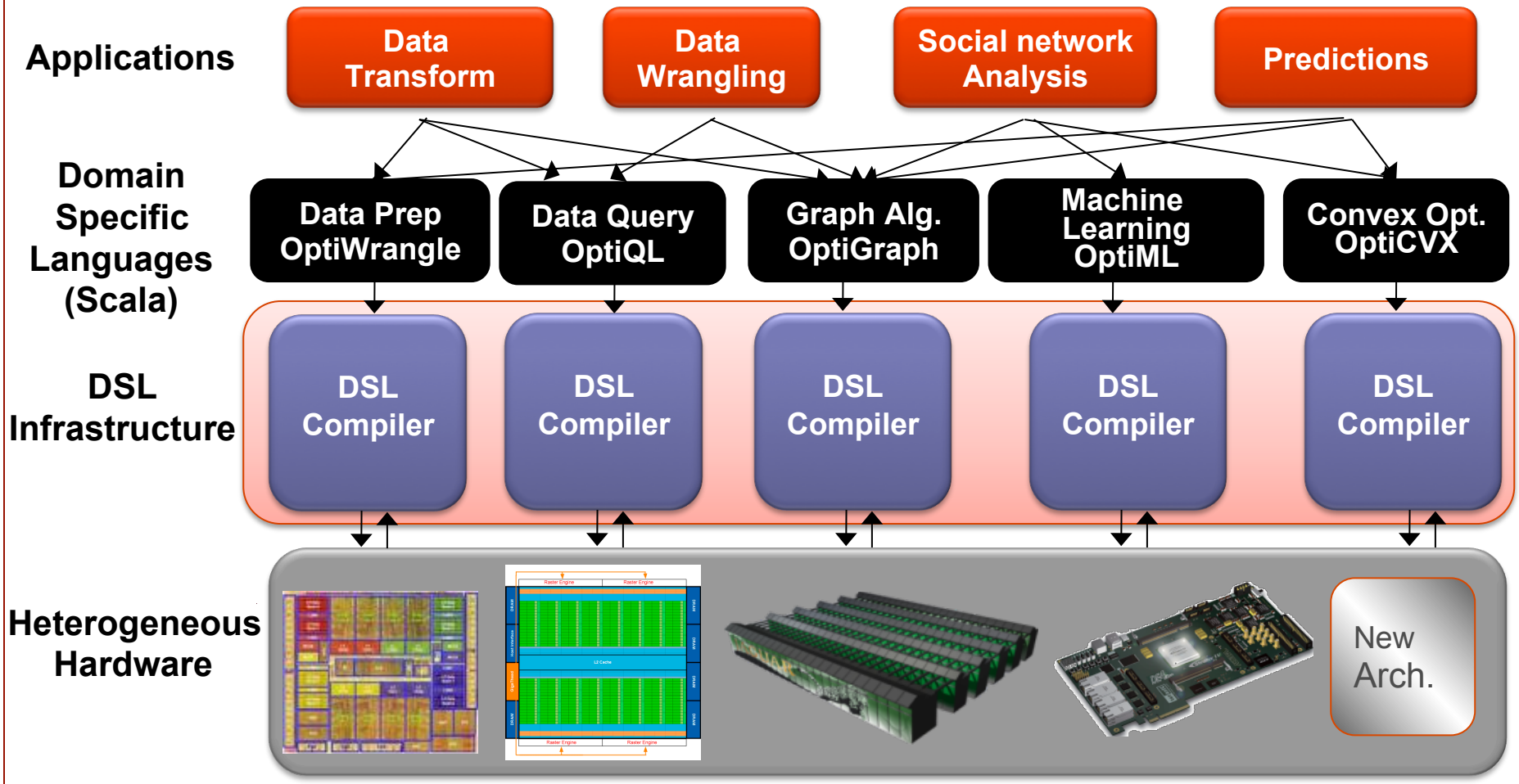
Portability and forward scalability

- DSL & Runtime can be evolved to take advantage of latest hardware features
- Applications remain unchanged
- Allows innovative HW without worrying about application portability

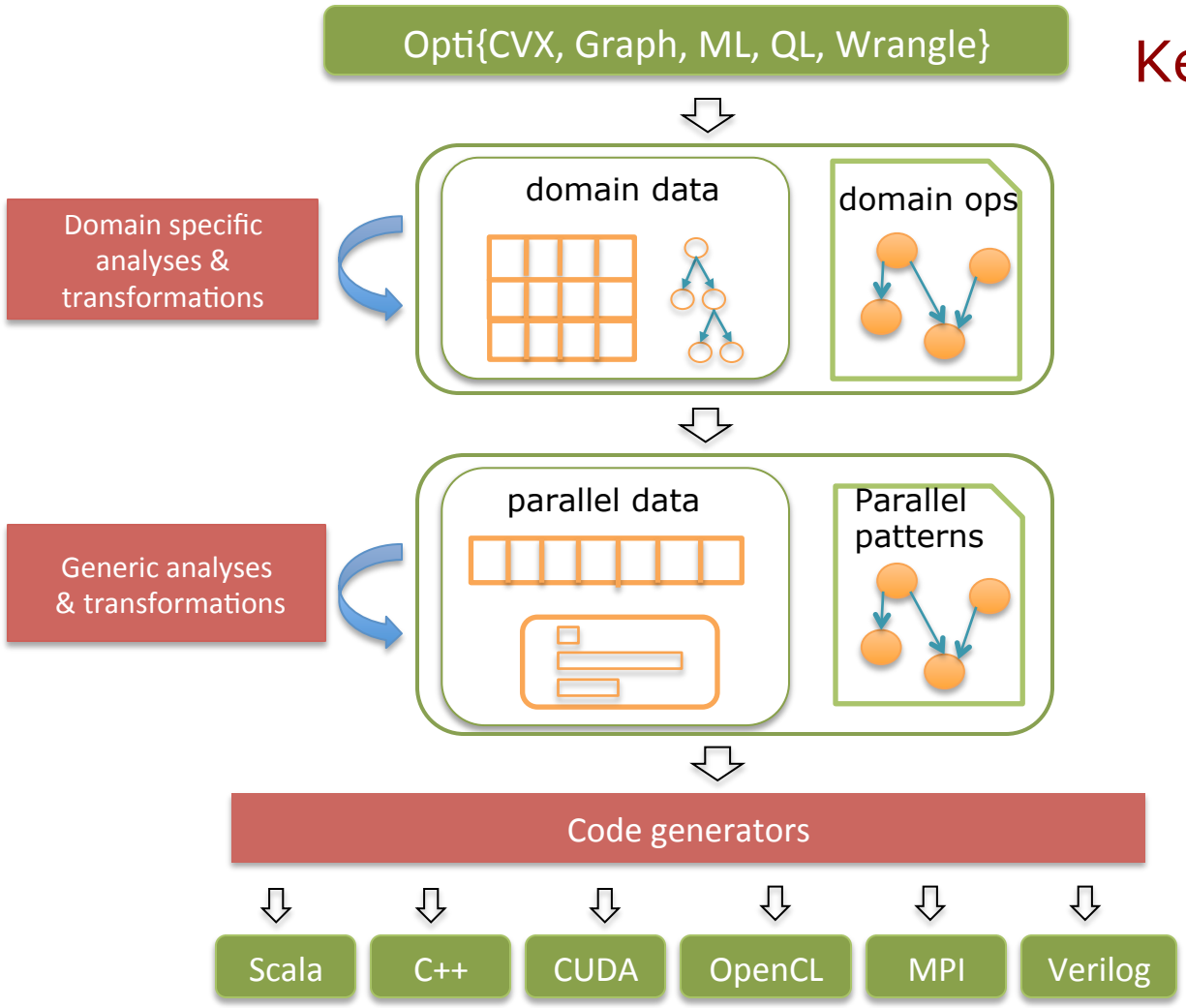
Our Approach: Data Analytics DSLs



Delite: DSL Infrastructure



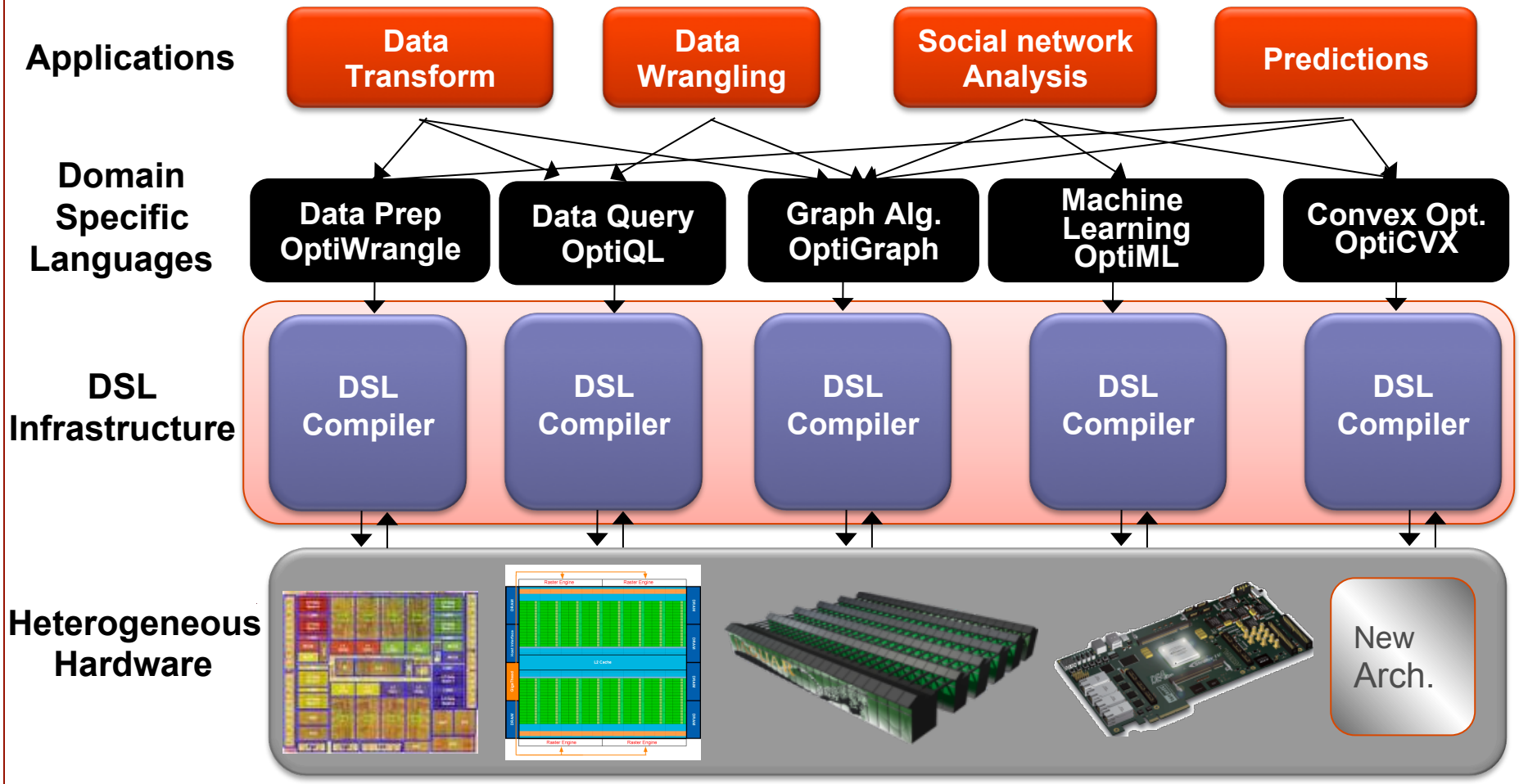
Delite Overview



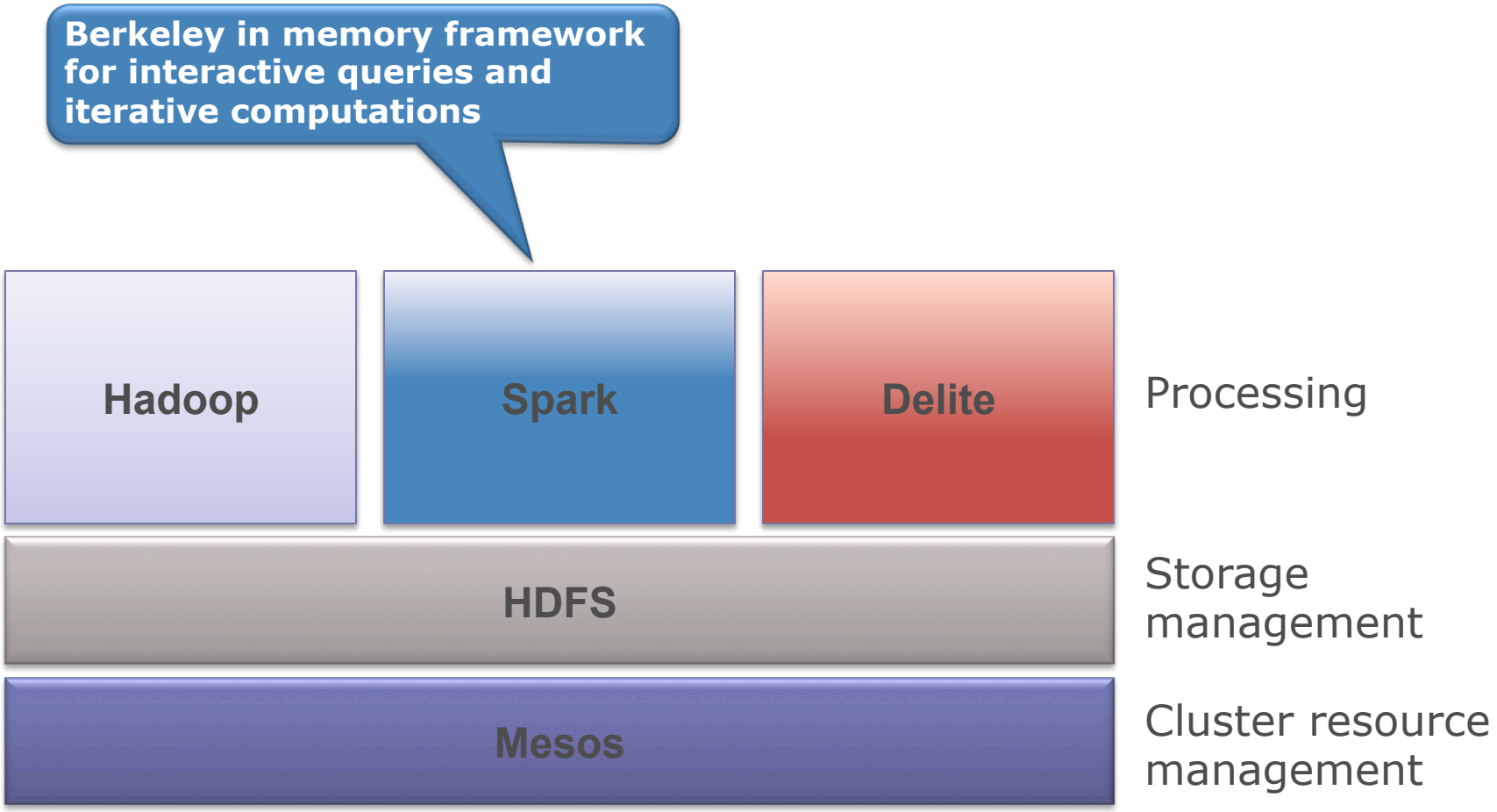
Key elements

- DSLs embedded in Scala
- IR created using staging
- Domain specific optimization
- General parallelism and locality optimizations
- Mapping to HW targets

Delite: DSL Examples



Big Data Analytics Systems

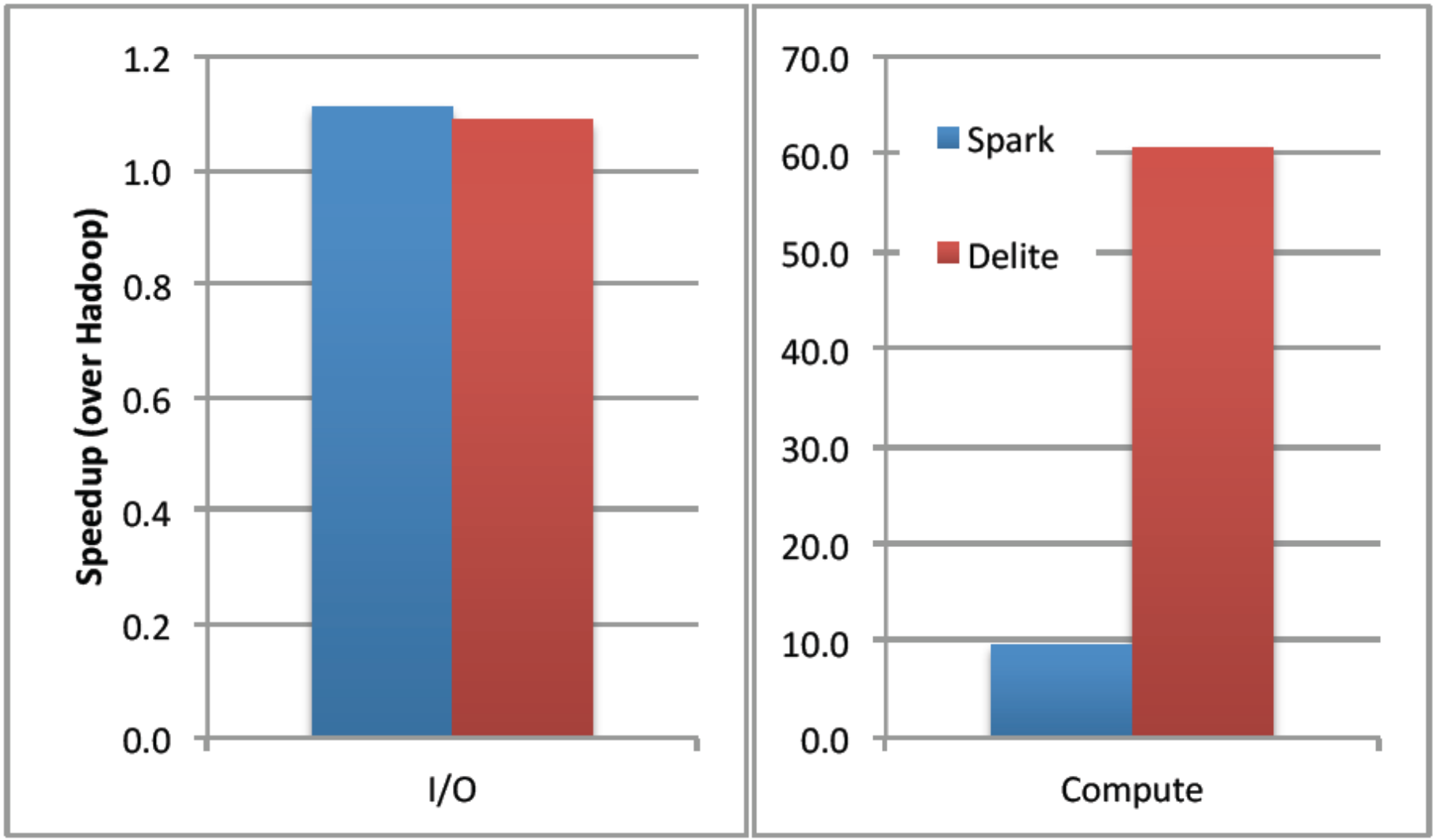


OptiQL

```
// lineItems: Table[LineItem]
val q = lineItems
  Where(_.l_shipdate <=
    Date("1998-12-01"))
  GroupBy(l => l.l_linestatus)).
  Select(g => new Result {
    val linestatus = g.key
    val sumQty = g.Sum(_.l_quantity)
    val sumDiscountedPrice =
      g.Sum(l => l.l_extendedPrice*
        (1.0-l.l_discount))
    val avgPrice =
      g.Average(_.l_extendedPrice)
     val countOrder = g.Count
  })
  OrderBy(_.returnFlag)
  ThenBy(_.lineStatus)
```

- In-memory data querying
- LINQ, SQL like
- Key operations are query operators on the `Table` data structure
  - User-defined schema
- Optimizations:
  - Fusion eliminates temporary allocations
  - Eliminate fields not used in query

TPC-H Query 1 on 20 x 4 cores

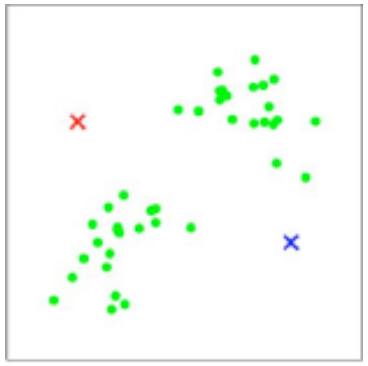


OptiML

OptiML: An Implicitly Parallel Domain-Specific Language for Machine Learning, **ICML 2011**

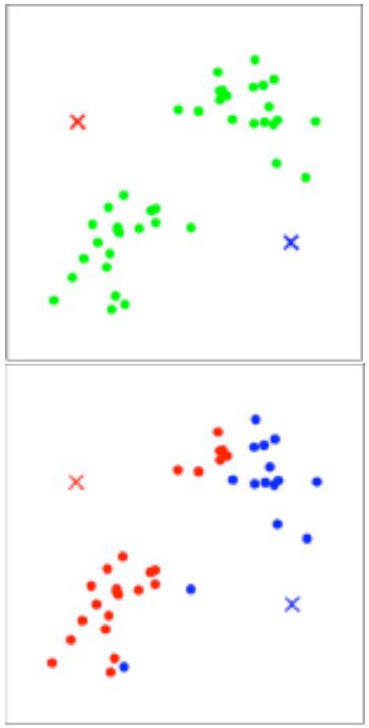
- Provides a familiar (MATLAB-like) language and API for writing ML applications
  - Ex. `val c = a * b` (a, b are Matrix[Double])
- Implicitly parallel data structures
  - Vector[T], Matrix[T], Stream[T]
  - `val c = (0::100) { i => i*2 }  // vector constructor`
- Implicitly parallel control structures
  - sum{…}, (0::end) {…}, gradient { … }, untilconverged { … }
  - Allow anonymous functions with restricted semantics to be passed as arguments of the control structures

OptiML: *k*-means Clustering



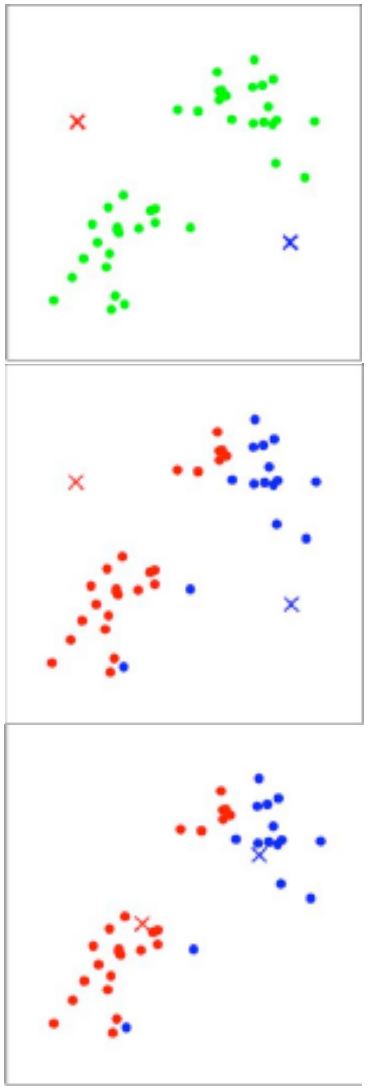
```
untilconverged(mu, tol){ mu =>
    // Find closest centroid to each sample




    // move each cluster centroid to the
    // mean of the samples assigned to it




}
```

OptiML: *k*-means Clustering



```
untilconverged(mu, tol){ mu =>
    // Find closest centroid to each sample
    val c = (0::m){i =>
       val allDistances = mu mapRows { centroid =>
          dist(samples(i), centroid)
       }
       allDistances.minIndex
    }

    // move each cluster centroid to the
    // mean of the samples assigned to it

}
```

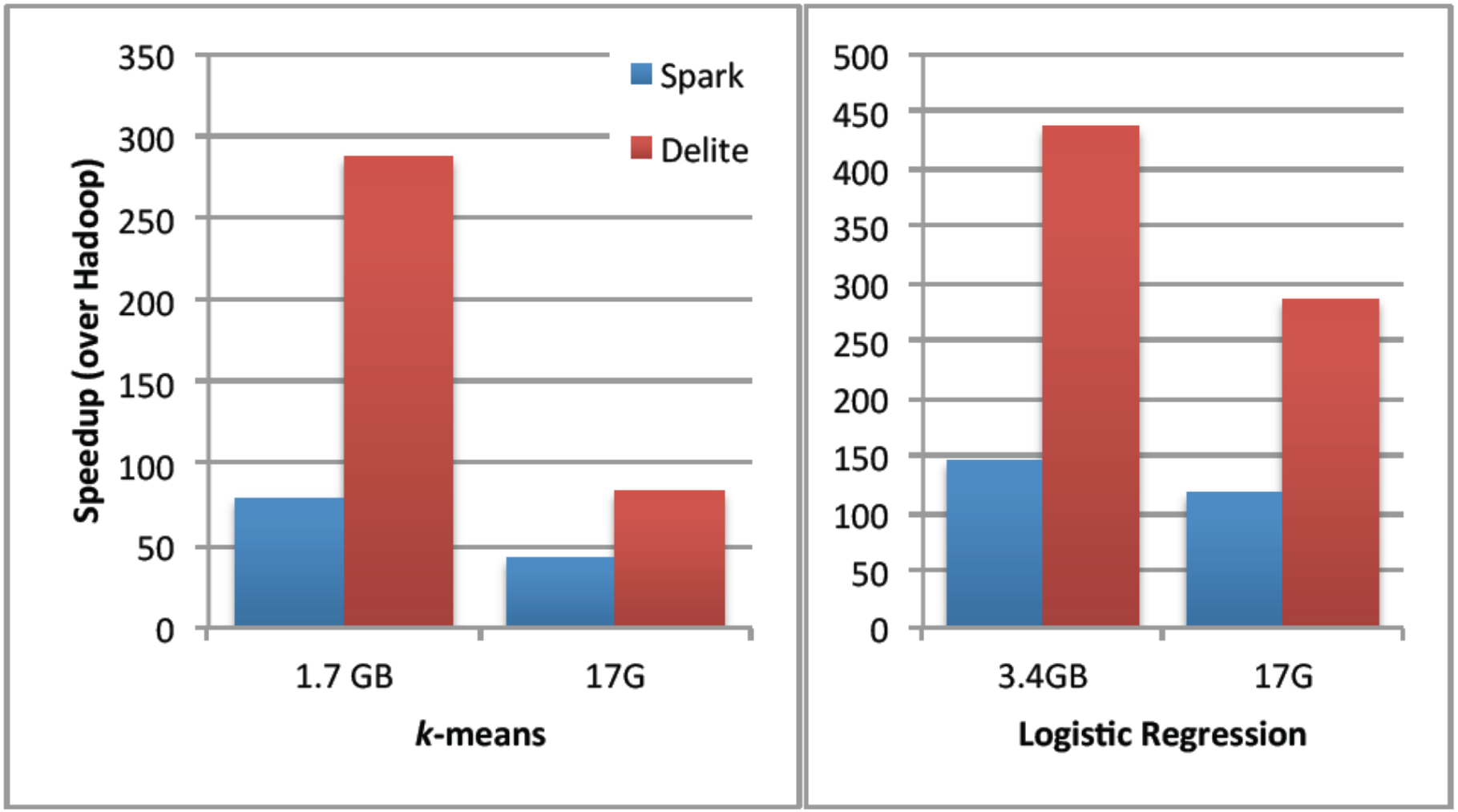
OptiML: *k*-means Clustering



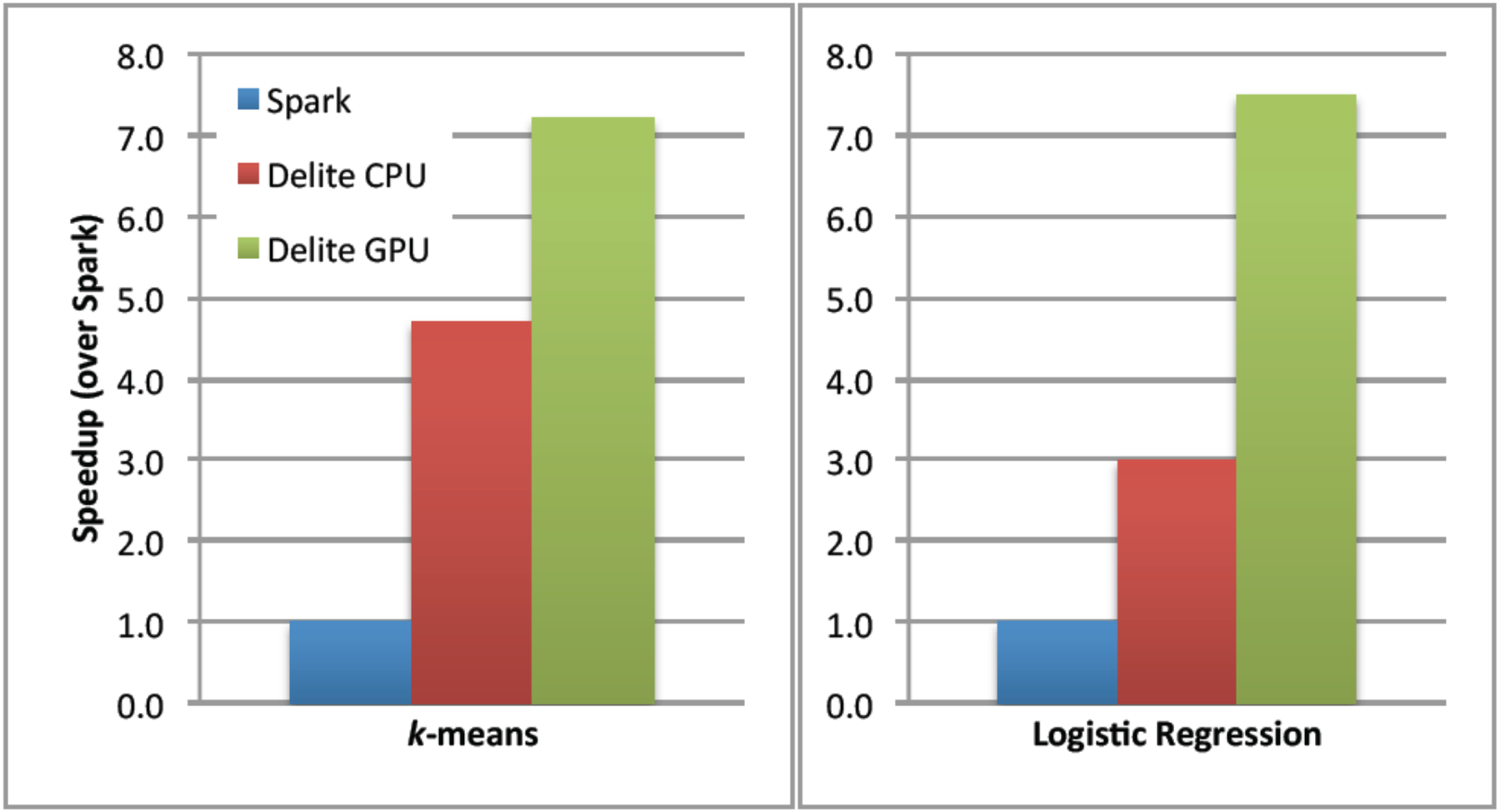
```
untilconverged(mu, tol){ mu =>
    // Find closest centroid to each sample
    val c = (0::m){i =>
       val allDistances = mu mapRows { centroid =>
          dist(samples(i), centroid)
       }
       allDistances.minIndex
    }

    // move each cluster centroid to the
    // mean of the samples assigned to it
    val newMu = (0::k,*){ cluster =>
       val weightedpoints =
         sumRowsIf(0,m)(i => c(i) == cluster){ i => samples(i) }
       val d = c.count(i => i == cluster)
       weightedpoints / d
    }
    newMu
}
```

- No map-reduce
- No key value pairs
- Efficient cluster implementation

Machine Learning on 20 x 4 cores: Library vs. Compiler



Machine Learning on 4 x 12 cores and 4 x GPU



OptiGraph

- A DSL for large-scale graph analysis based on Green-Marl
  - A DSL for Real-world Graph Analysis
  - Green-Marl: A DSL for Easy and Efficient Graph Analysis (Hong et. al.), ASPLOS '12
- Data structures
  - Graph (directed, undirected), node, edge,
  - Set of nodes, edges, neighbors, …
- Graph iteration
  - Normal parallel iteration, Breadth-first iteration, Topological Order, …
- Deferred assignment and parallel reductions (Bulk synchronous consistency)

OptiGraph: PageRank

Implicitly parallel iteration

```
for(t <- G.Nodes) {
  val rank = ((1.0 d)/ N) +
             d * Sum(t.InNbrs){w => PR(w) / w.OutDegree}
  PR <= (t,rank)
  diff += Math.abs(rank - PR(t))
}
```

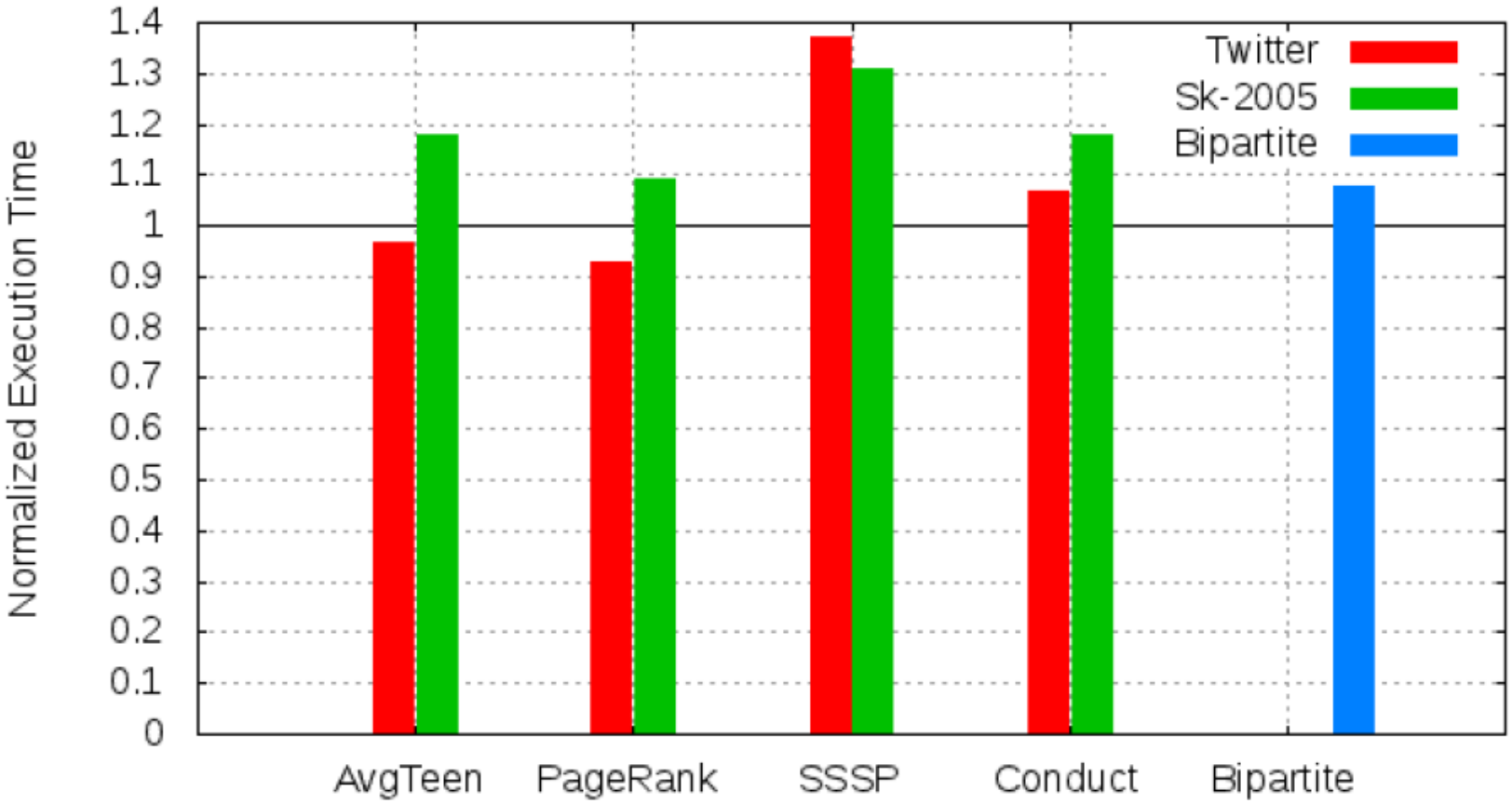
Deferred assignment and scalar reduction

Writes become visible after the loop completes

Green-Marl vs. GPS (Pregel): Lines of Code

Algorithm	Green-Marl	Native GPS
Average Teenage Follower (AvgTeen)	13	130
PageRank	19	110
Conductance (Conduct)	12	149
Single Source Shortest Paths (SSSP)	29	105
Random Bipartite Matching (Bipartite)	47	225
Approximate Betweeness Centrality	25	Not Available

Green-Marl vs. GPS (Pregel) on 20 x 4 cores



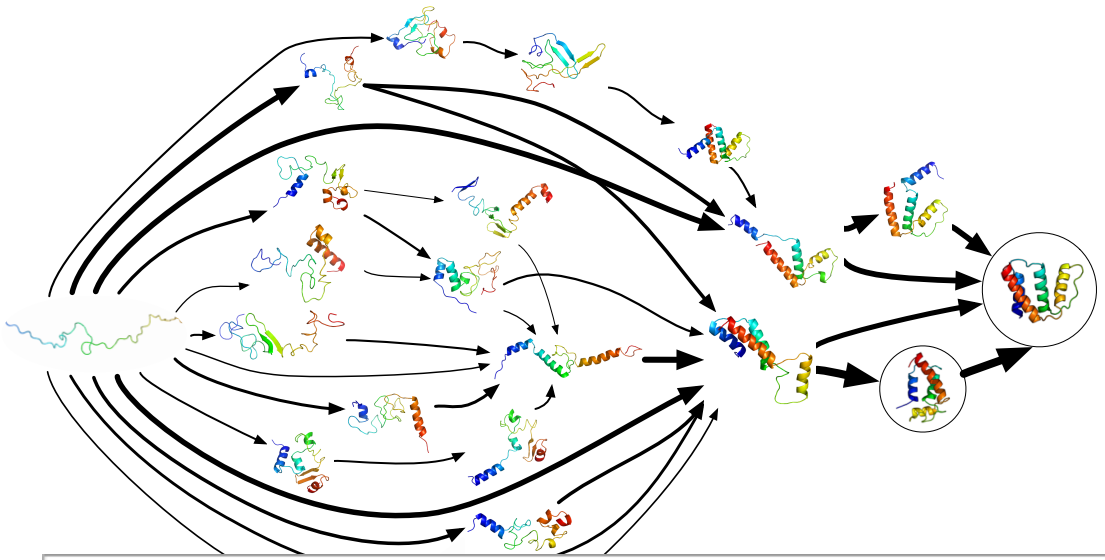
Conclusions

- DSLs are the key to next generation big data analytics
  - High Productivity: higher level abstractions
  - High performance: fine-grained parallelism
- Sophisticated compilers needed to make sense of high-level, domain-specific abstractions
- Performance advantage of compiling DSLs is substantial
- http://ppl.stanford.edu

DSLs: Barriers to High Performance

- Problem 1: abstraction penalty
  - Staging: remove abstraction programmatically using partial evaluation
- Problem 2: compiler lacks semantic knowledge
  - Extend compiler with high-level knowledge
    - E.g. Teach compiler linear algebra
- Problem 3: compiler lacks parallelism knowledge
  - Extend the compiler with parallelism and locality knowledge
- Solving any of the problems alone will not result in high performance

MSM Builder Using OptiML
with Vijay Pande



Markov State Models (MSMs)

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