DryadLINQ

by Yuan Yu et al., OSDI'08

Ilias Giechaskiel

Cambridge University, R212
ig305@cam.ac.uk

January 28, 2014

Conclusions

Takeaway Messages

- SQL cannot express iteration
  - Unsuitable for machine learning, graph processing, etc.
- MapReduce cannot express `Join`
  - Also, simplistic, so no automatic optimizations
- DryadLINQ fills the void:
  - Define declarative-imperative programming model using LINQ
  - Automatically and transparently optimize and distribute
  - Execute on top of Dryad infrastructure

Context: Language

LINQ [MBB06]

- Language-Integrated Query
- Design pattern of standard query operators
- SQL-like syntax + lambda expressions and anonymous types
- C#, F#, VB implementations
- Part of .NET development framework

Context: Platform

Dryad [IBY+07]

- "General-purpose distributed execution engine"
- Dataflow DAG graph (developer-provided)
  - Vertices: sequential programs
  - Edges: communication channels
  - Dynamic
- Dryad engine handles
  - Scheduling
  - Recovery
  - Data transfer

Dryad Graph

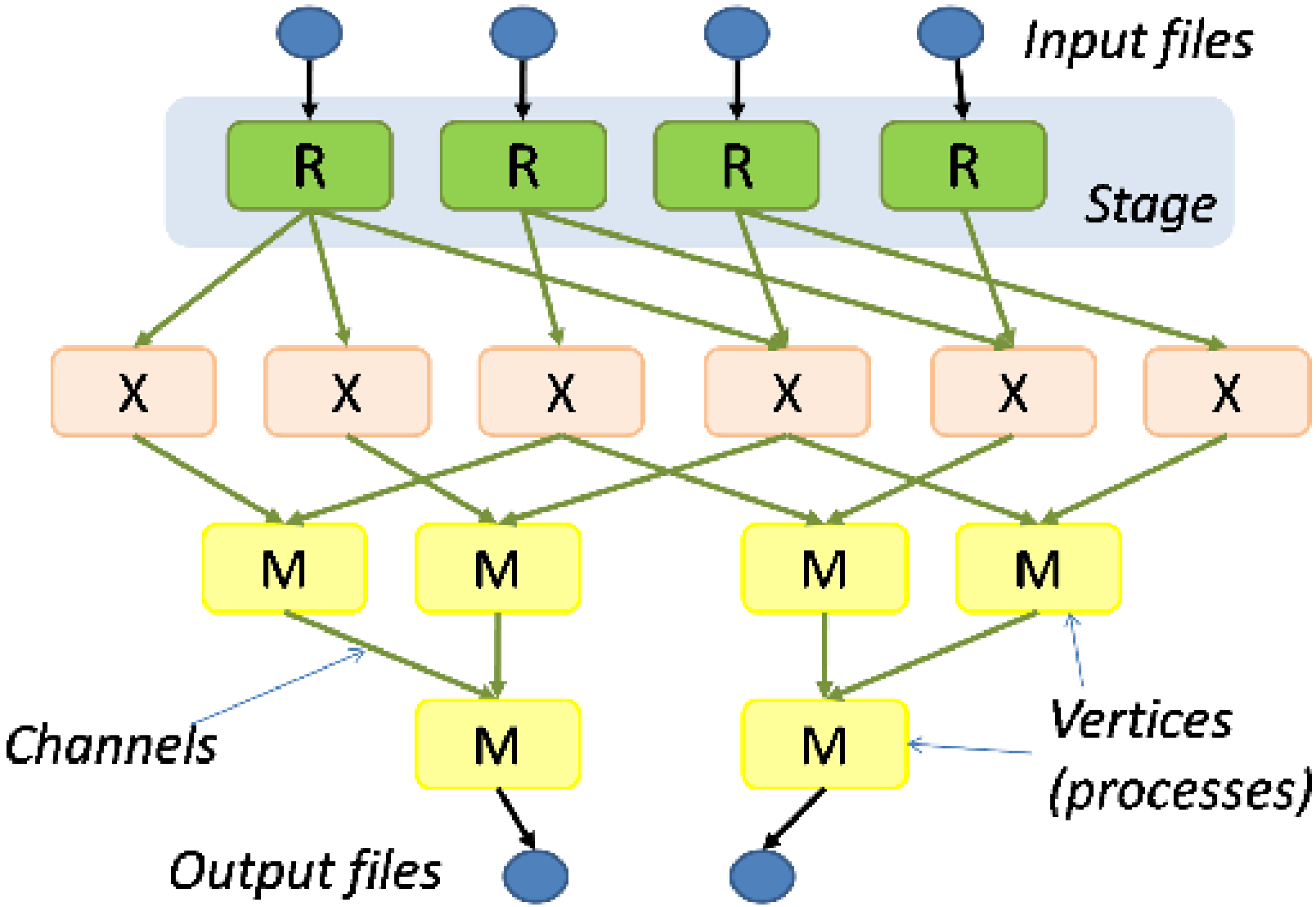


Figure: `https://research.microsoft.com/en-us/projects/dryad/`

Software Layers

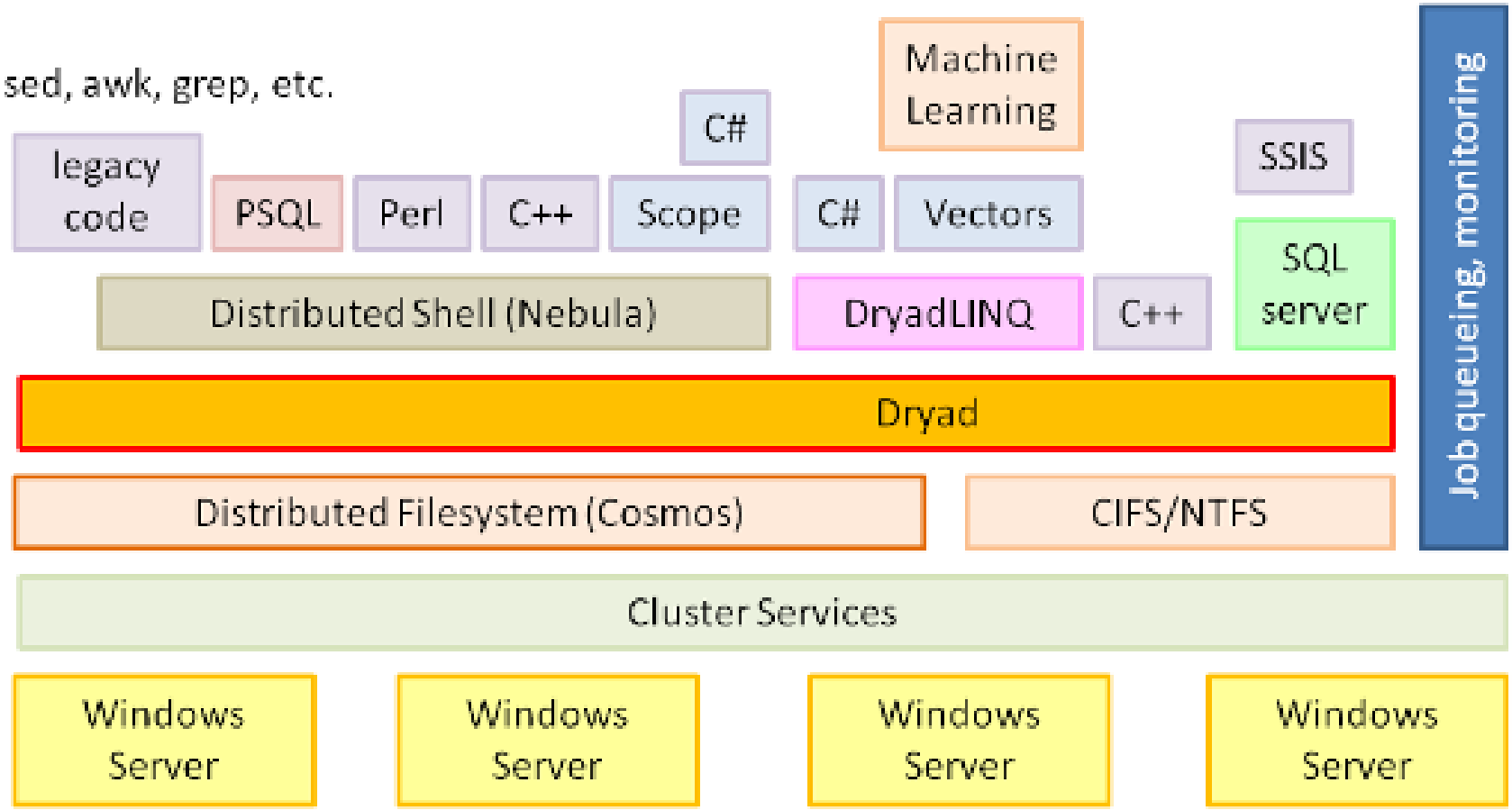


Figure: `https://research.microsoft.com/en-us/projects/dryad/`

DryadLINQ: Motivation

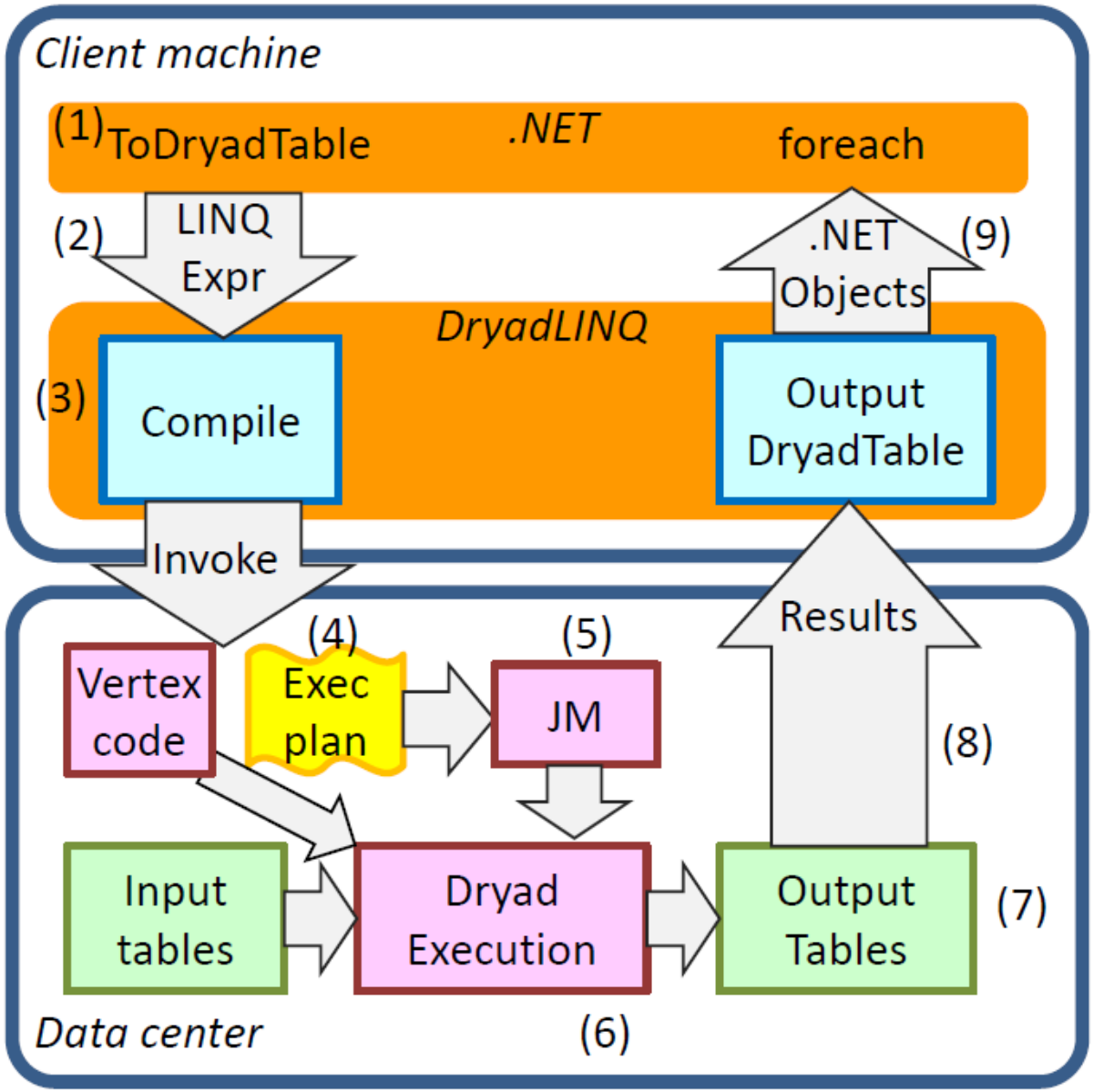
DBMS vs. MapReduce [PPR+09]

- Parallel DBMS
  - Robust, highly available
  - Faster and less code
  - Longer to tune and load data
  - Insufficient expressiveness
- MapReduce
  - Popular, simple
  - Less expressive and general

DryadLINQ

- Best of both worlds using LINQ on Dryad
- Hide Dryad complexity by automatic DAG construction
- Automatic scheduling and optimizations
- Transparent dynamic changes

DryadLINQ: Execution



DryadLINQ: Histogram Example [YIF+08a]

```
public static IQueryable<Pair>
    Histogram(IQueryable<string> input, int k)
{
IQueryable<string> words = input.SelectMany(x
    => x.Split(' '));
IQueryable<IGrouping<string, string>> groups =
    words.GroupBy(x => x);
IQueryable<Pair> counts = groups.Select(x =>
    new Pair(x.Key, x.Count()));
IQueryable<Pair> ordered =
    counts.OrderByDescending(x => x.count);
IQueryable<Pair> top = ordered.Take(k);
return top;
}
```

DryadLINQ: Histogram Example [YIF+08a]

```
"A line of words of wisdom"
        SelectMany(x => x.Split(' '));
["A","line","of","words","of","wisdom"]
        GroupBy(x => x);
[["A"],["line"],["of","of"],["words"],["wisdom"]]
        Select(x => new Pair(x.Key, x.Count()));
[{"A",1}, {"line",1}, {"of",2}, {"words",1},
   {"wisdom",1}]
        OrderByDescending(x => x.count);
[{"of",2}, {"A",1}, {"line",1}, {"words", 1},
   {"wisdom",1}]
        Take(3);
[{"of", 2}, {"A", 1}, {"line", 1}]
```

DryadLINQ: Optimizations

Static

- Conditional graph rewriting rules
- Pipelining
- Removing redundancy
- Eager aggregation
- I/O reduction

Dynamic

- During Dryad job execution
- Hooks in Dryad API
- Bases decisions on runtime topology

DryadLINQ: Experimental Evaluation

Hardware Configuration

- 240 computers
  - Two 2.6GHz dual-core AMD CPUs
  - 16GB RAM
  - Four 750GB SATA drives
- Connected through Linksys 48-port GBit Ethernet switches

Benchmarks

- Terasort
- SkyServer
- PageRank
- Large-Scale Machine Learning

DryadLINQ: Experimental Evaluation

Conclusions

- TeraSort
  - Constant average performance on local switches
  - Asymptotic behavior for more than one switch
- SkyServer
  - DryadLINQ fewer LOC than Dryad, but:
  - 1.3 times slower!
- PageRank
  - Optimized implementation 18x faster than naive version
- ML
  - Algorithms 50x faster than single computer

Evaluation

Criticisms

- Debugging
  - No-side-effect rule neither checked nor enforced
  - Easy to re-execute vertex, but what vertex?
  - Performance debugging harder
  - Job visualization [JYB11]
- Programming
  - Complex statements need annotations
  - LINQ syntax
- Performance
  - Lack of comparison with different systems
  - Lack of incremental processing
  - Early prototype

DryadLINQ: Today

Current State

- Spawned DryadInc for incremental computations [PBYI09]
- Dryad has been abandoned in favor of Hadoop
  `msdn.microsoft.com/en-us/library/hh378101.aspx`
- Naiad was started to address incremental shortcomings
  `research.microsoft.com/en-us/projects/naiad/`
  - Dataflow and graph ideas remain
  - New model developed

Conclusions

Key Insights

- Benefit from both DBMS and MapReduce
  - Hybrid programming style in known environment
- Combine static heuristics and runtime optimizations
- Give the illusion of single thread
  - Make distribution transparent to programmer

Key Questions

- How can you debug distributed applications?
- How does DryadLINQ compare to other platforms?
  - Performance and program implementation
- How can you optimize for incremental computations?
- Why was Dryad abandoned?
- Your questions?

Bibliography I


Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly, *Dryad: Distributed data-parallel programs from sequential building blocks*, Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007 (New York, NY, USA), EuroSys '07, ACM, 2007, pp. 59–72.


Vilas Jagannath, Zuoning Yin, and Mihai Budiu, *Monitoring and debugging dryadlinq applications with daphne*, Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (Washington, DC, USA), IPDPSW '11, IEEE Computer Society, 2011, pp. 1266–1273.

Bibliography II


Erik Meijer, Brian Beckman, and Gavin Bierman, *Linq: Reconciling object, relations and xml in the .net framework*, Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (New York, NY, USA), SIGMOD '06, ACM, 2006, pp. 706–706.


Lucian Popa, Mihai Budiu, Yuan Yu, and Michael Isard, *Dryadinc: Reusing work in large-scale computations*, Proceedings of the 2009 Conference on Hot Topics in Cloud Computing (Berkeley, CA, USA), HotCloud'09, USENIX Association, 2009.

Bibliography III

Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker, *A comparison of approaches to large-scale data analysis*, Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (New York, NY, USA), SIGMOD '09, ACM, 2009, pp. 165–178.

Yuan Yu, Michael Isard, Dennis Fetterly, Mihai Budiu, Ú Erlingsson, Pradeep Kumar Gunda, Jon Currey, Frank McSherry, and Kannan Achan, *Some sample programs written in dryadlinq*, Tech. report, 2008.

Bibliography IV


Yuan Yu, Michael Isard, Dennis Fetterly, Mihai Budiu, Úlfar Erlingsson, Pradeep Kumar Gunda, and Jon Currey, *Dryadlinq: A system for general-purpose distributed data-parallel computing using a high-level language*, Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation (Berkeley, CA, USA), OSDI'08, USENIX Association, 2008, pp. 1–14.