DryadLINQ

A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language

Arman Idani

14 Feb 2012

R202 – Data Centric Networking

Background

- Major Distributed Computing Frameworks
  - MapReduce
  - Dryad
  - Apache Hadoop (open source MapReduce)

Motivation

- Internet-scale Services
  - Computationally intensive
  - Huge I/O (terabyte-scale)

- Datacenters
  - Thousands of servers
  - Commodity off-the-shelf hardware
  - They fail

Solution?

- Faster servers
  - Performance not scaling with computational need
  - Memory and I/O limits
- GPUs
  - Tied to underlying hardware implementation
  - Memory and I/O limits
- Parallel databases
  - Designed only for relational algebra manipulations

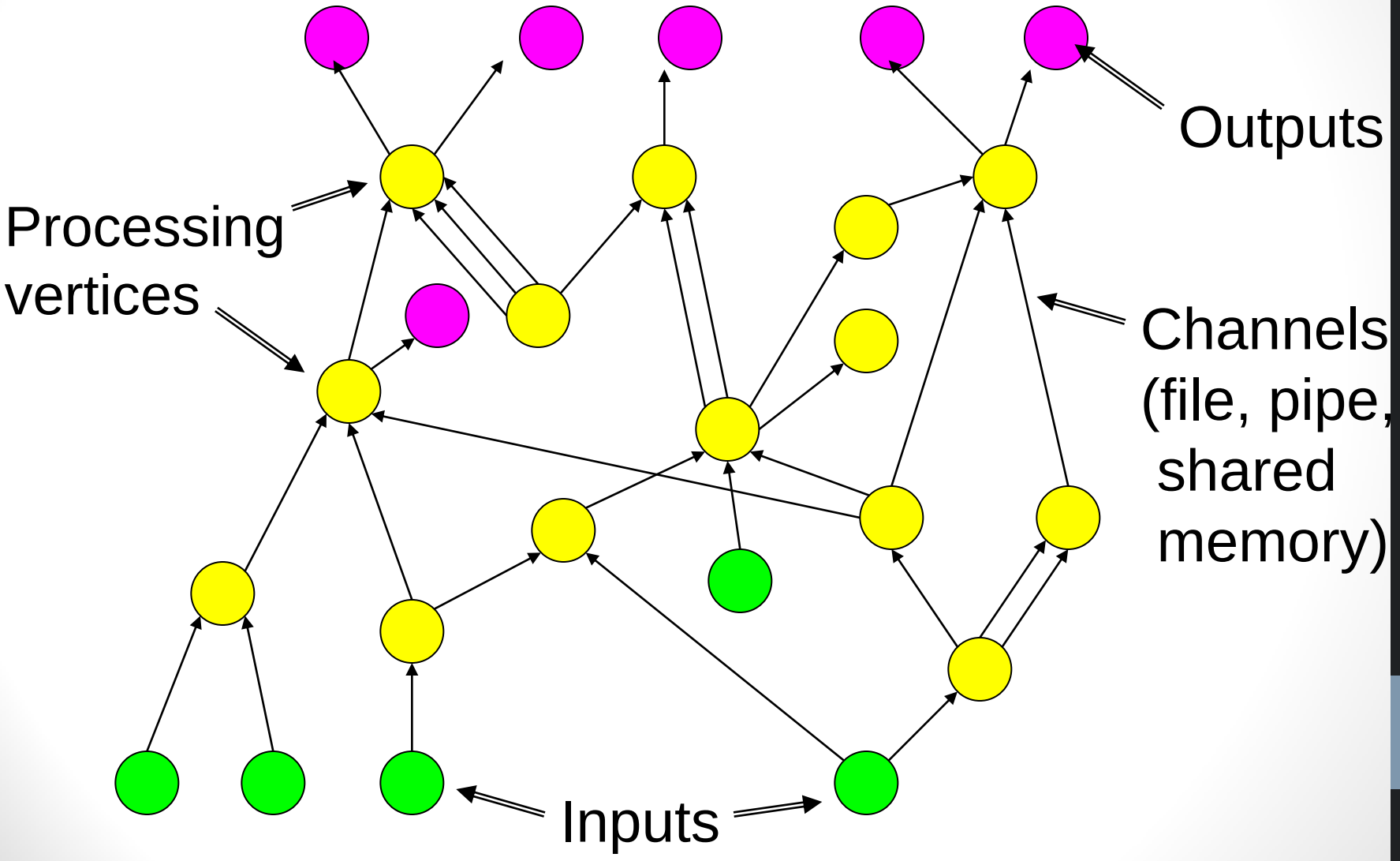
MapReduce

- Map and Reduce… that's it.
- No fault tolerance between Map and Reduce
- Reducers write to redundant storage
  - 2 network copies, 3 disk copies
- Architectural limits
  - No support for different types of I/O
- Ugly to program!

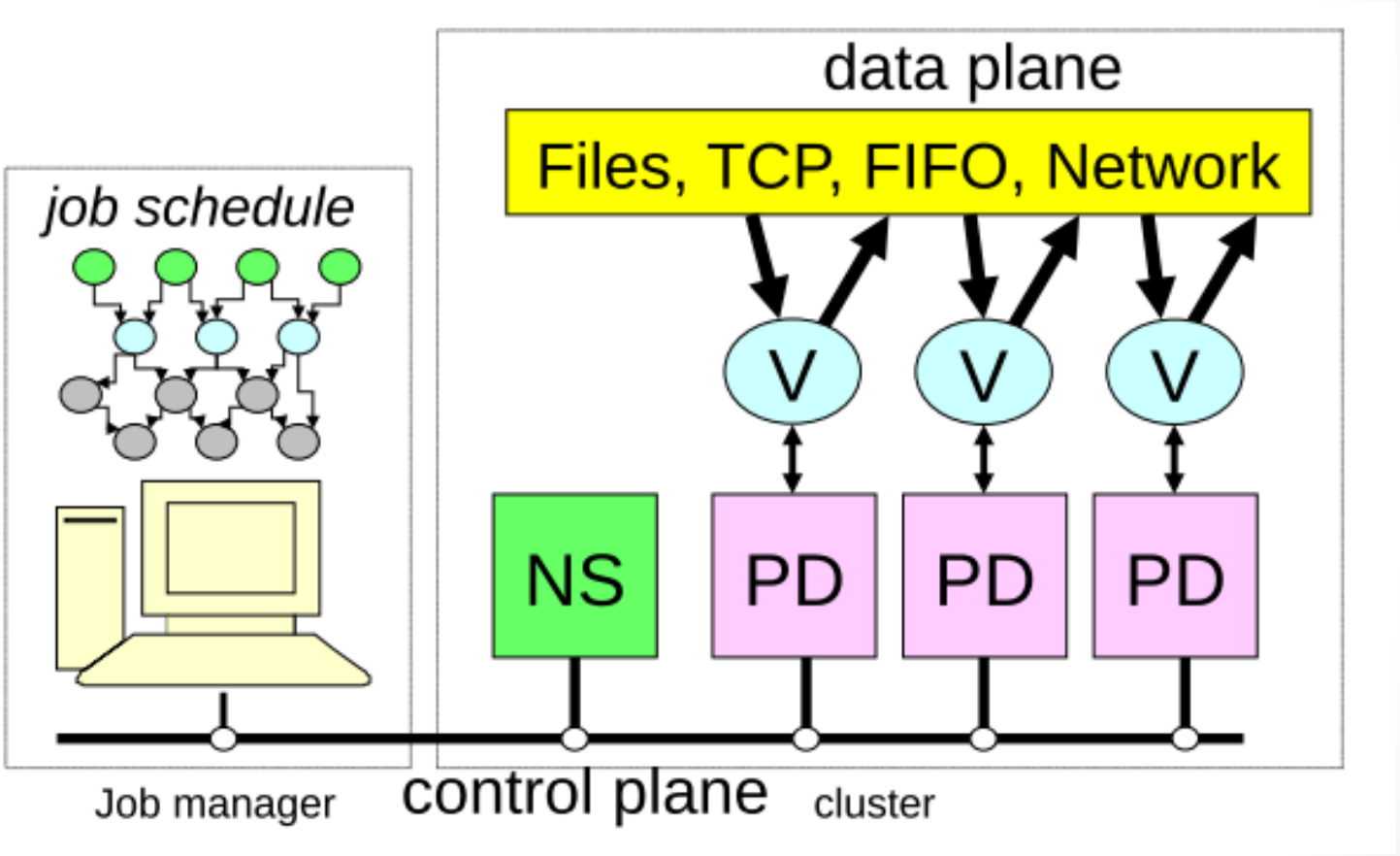
Dryad

- Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks (original paper)
- User defines dataflow of the program

Job = Directed Acyclic Graph



Dryad Architecture



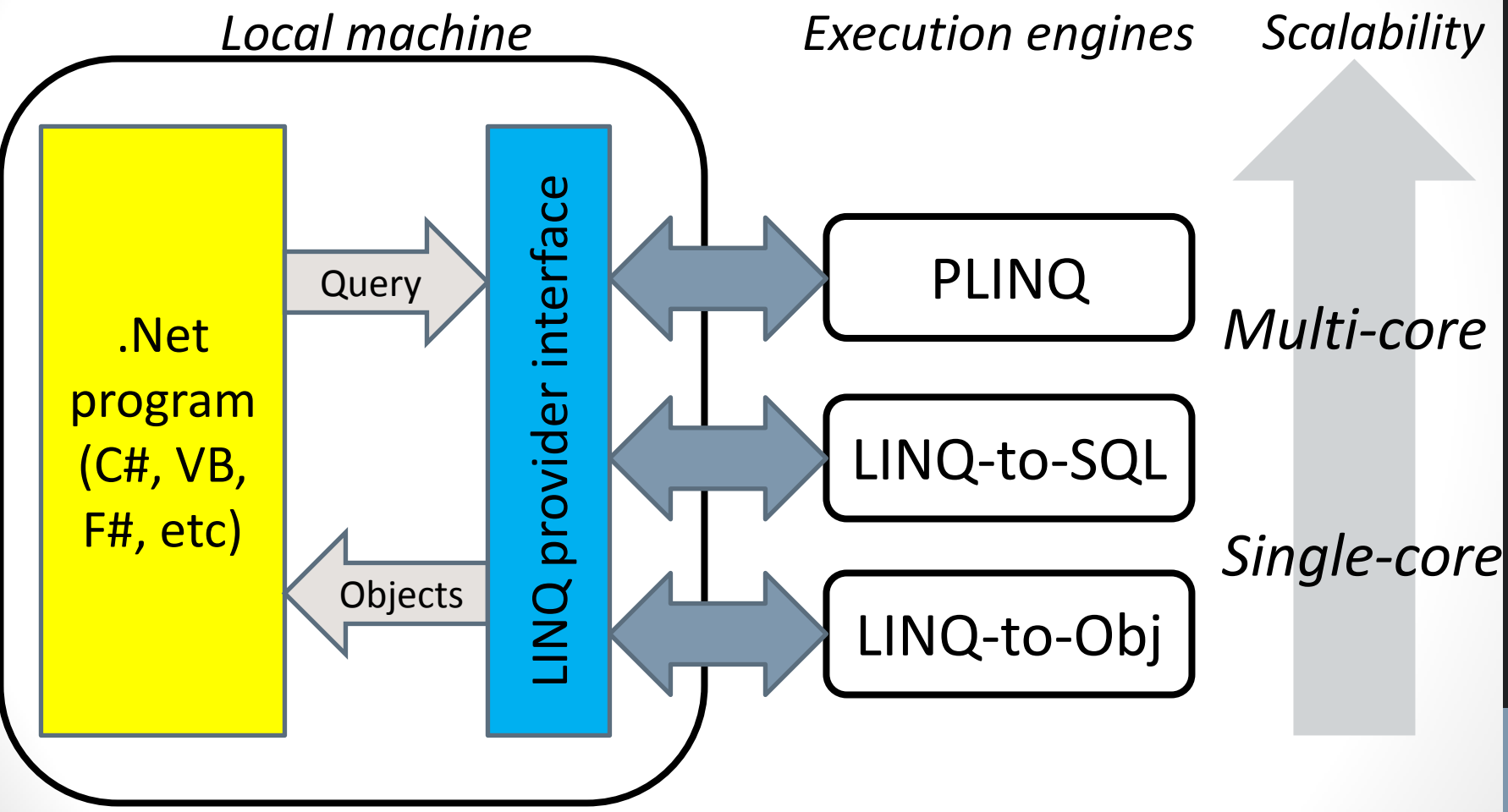
Dryad Properties

- Channel types
  - File transfer, Shared memory FIFO, TCP pipe
- Encapsulation
  - Convert a graph into a vertex for more complicated systems
- Fault tolerance for both vertices and inputs
  - Runs upstream vertices recursively if inputs are gone
- Map and Reduce classes
  - Easy to port MapReduce applications

LINQ

- Language INtegrated Query
  - A set of operators to manipulate datasets in .NET
  - All relational operators are supported
  - Integrated into C#, VB and F#
  - Declarative and Imperative programming
  - .NET development tools

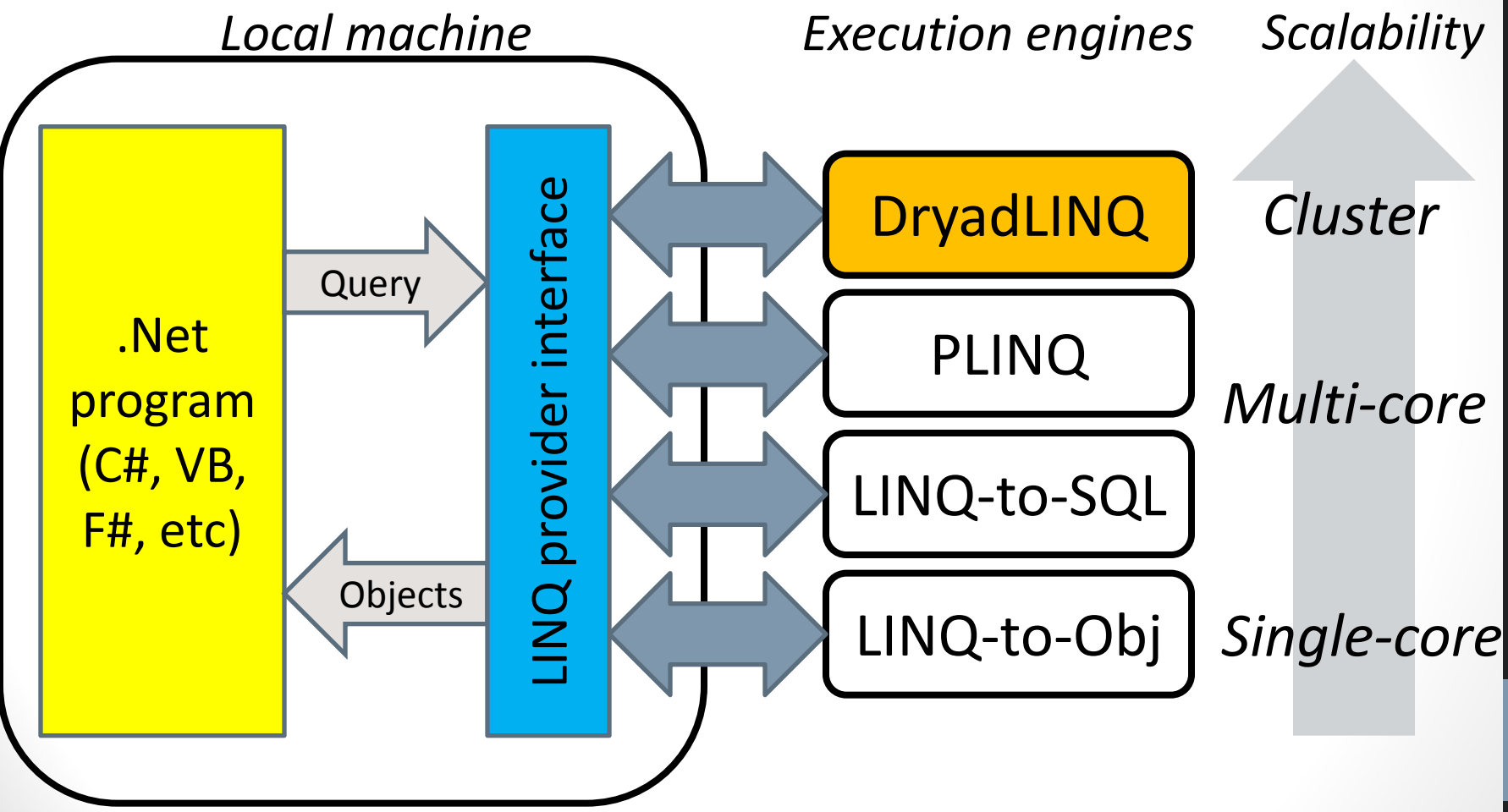
LINQ Architecture



DryadLINQ = Dryad + LINQ

- Problem: How to easily write distributed data-parallel programs for a computer cluster?
- Answer: Give the programmer the illusion of developing for a single computer
  - Let the system deal with parallelism and its complexities
  - Dryad: an execution engine for LINQ

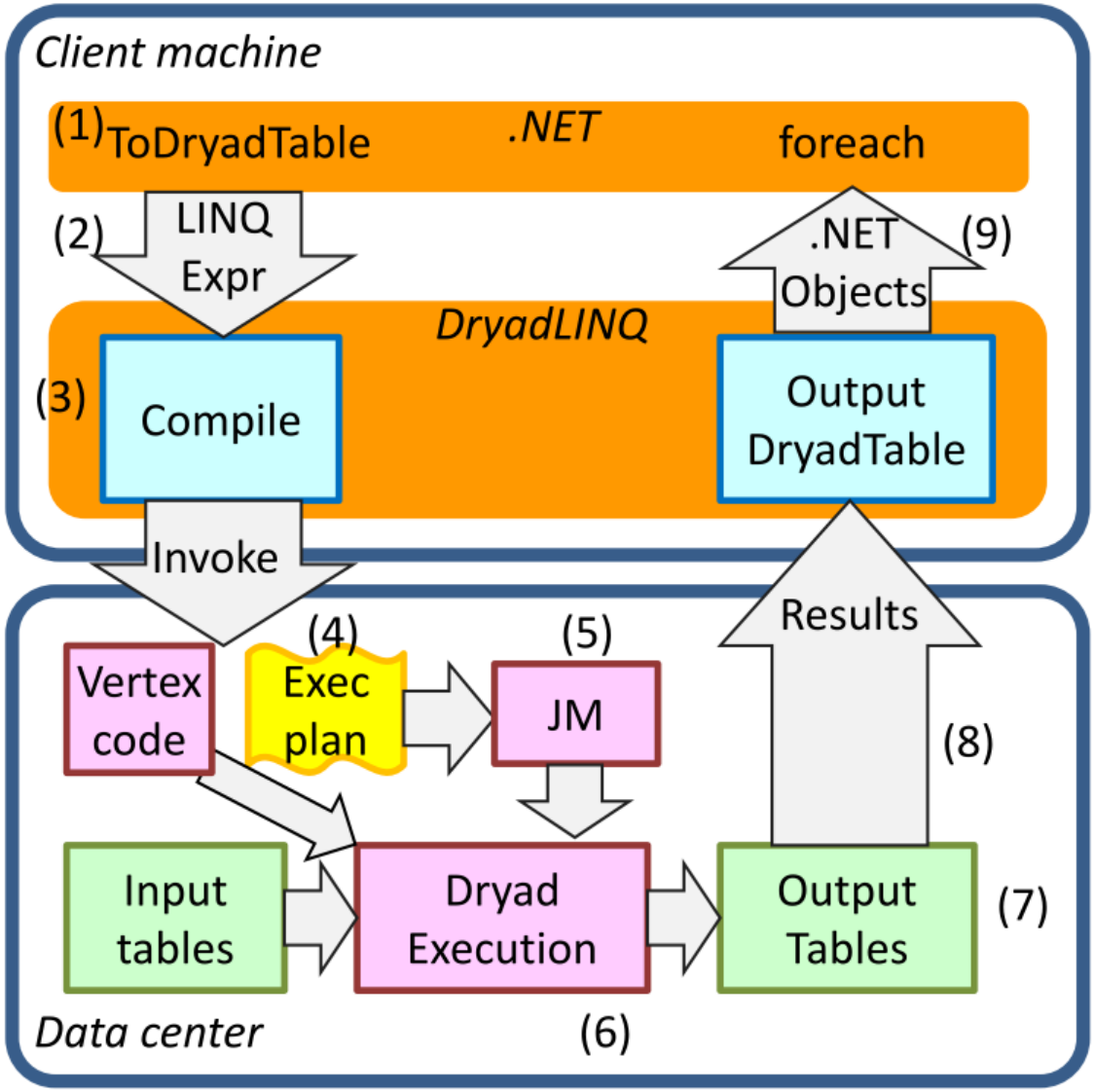
Dryad as LINQ's execution engine



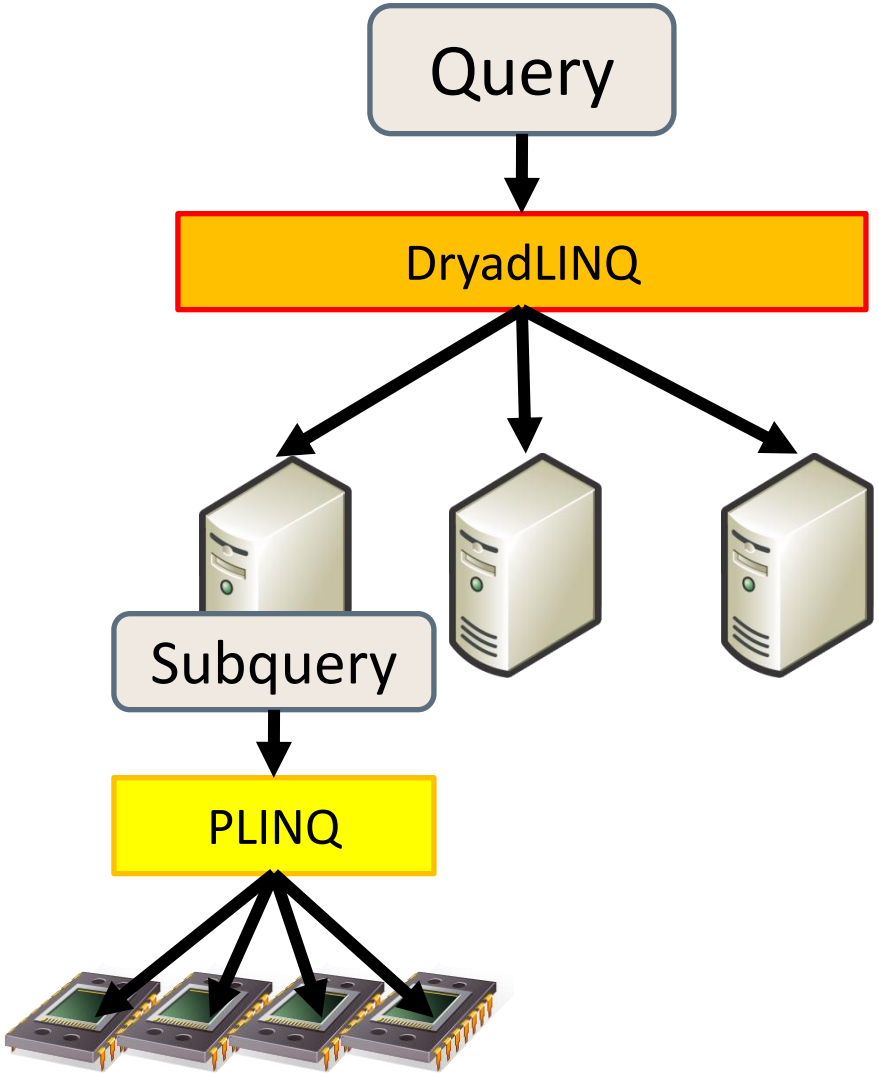
DryadLINQ

- Sequential, single machine programming abstraction
- Program runs on single-core, multi-core and a cluster
- Development in familiar programming languages
- Visual Studio development environment

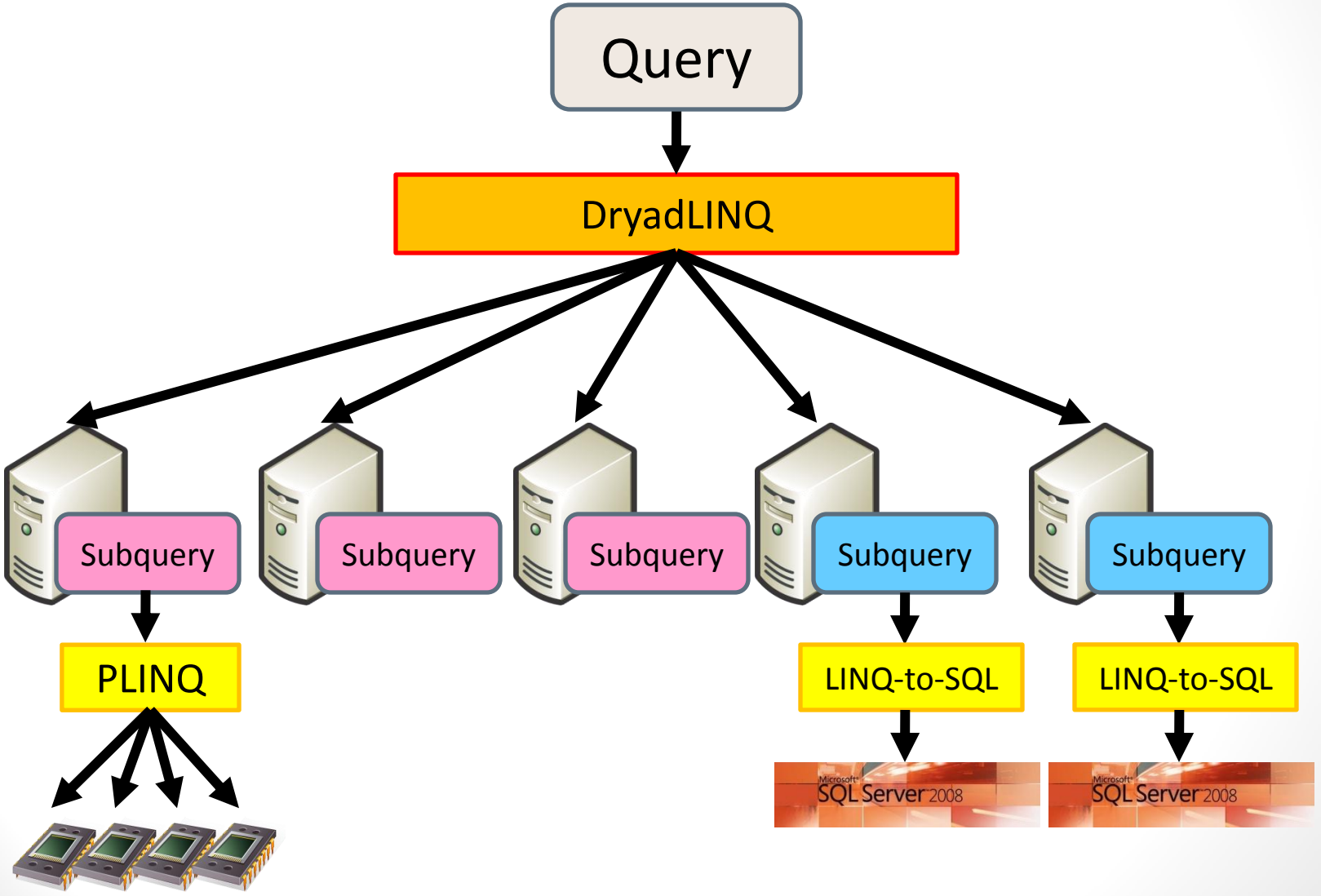
DryadLINQ Overview



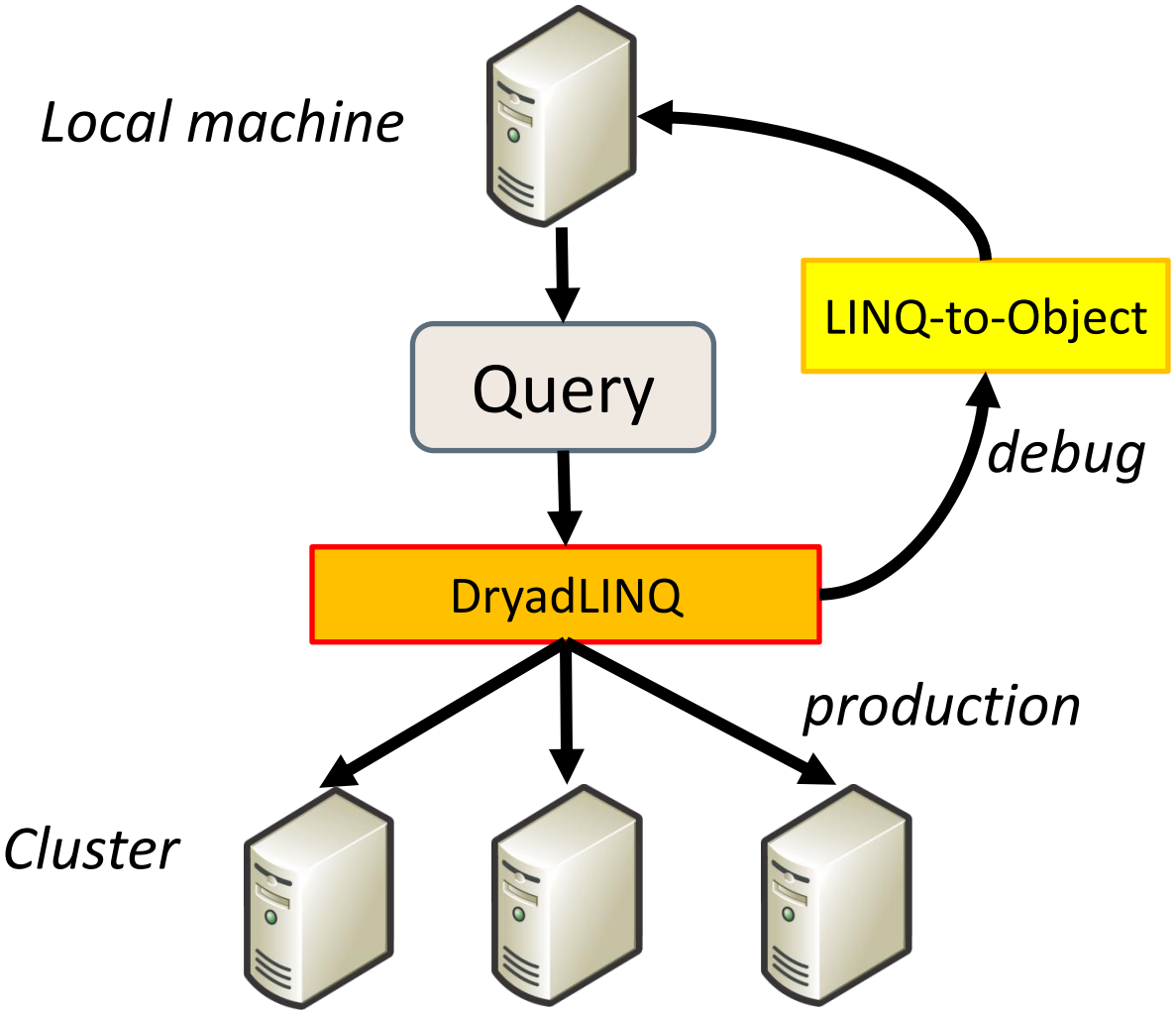
DryadLINQ LINQ Integration



DryadLINQ SQL Integration



DryadLINQ Local Simulation



Evaluation

- Configuration: 240 clusters (8x30)
  - Two dual-core AMD Opteron processors
  - 16GB of DDR2 RAM
  - Four stripped 750GB disks
- Benchmarks
  - TeraSort
  - SkyServer
  - PageRank
  - Machine Learning

TeraSort

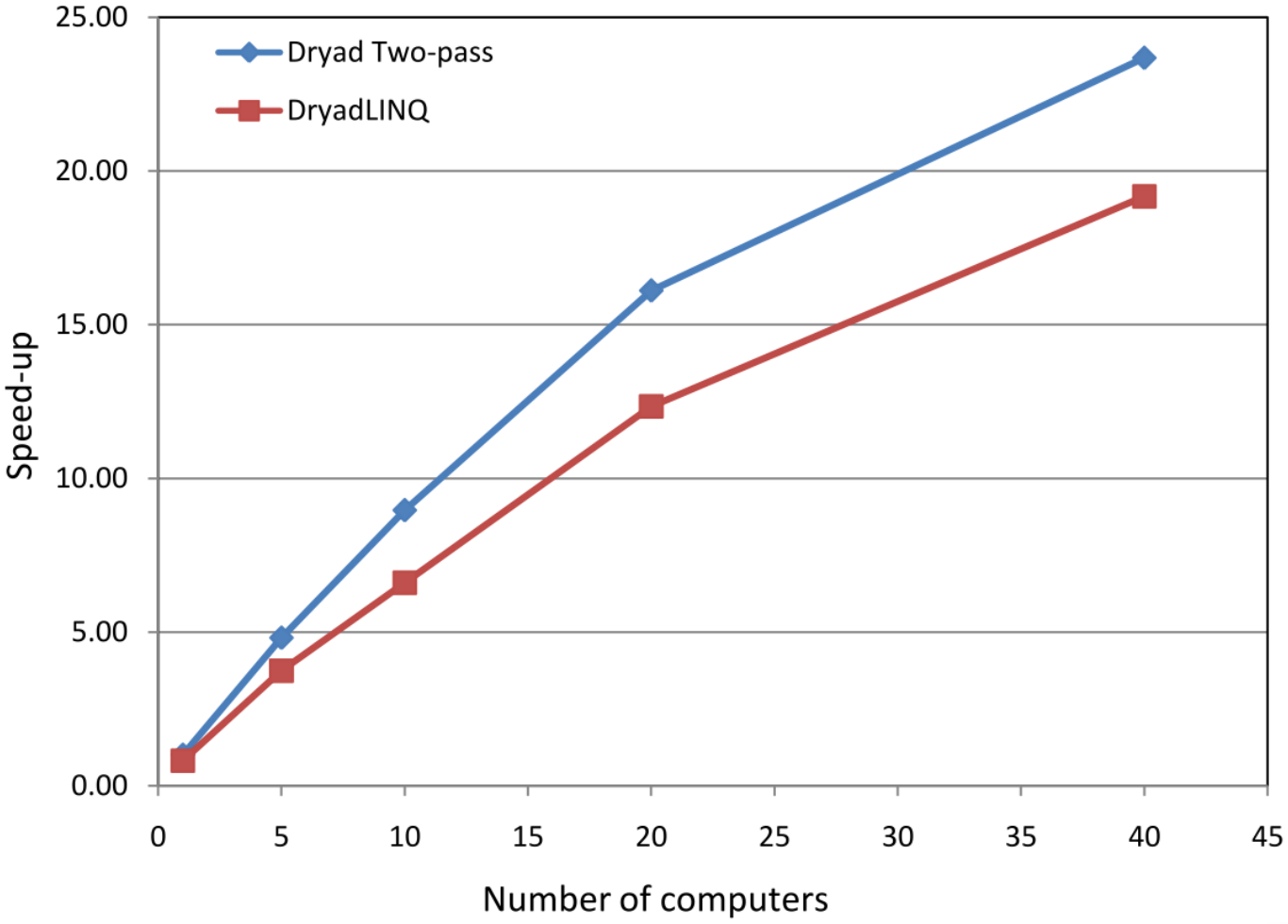
- Performance scaling (1 < n < 240)
- Sorting records by string comparisons
- Each node stores 3.87GB

Computers	1	2	10	20	40	80	240
Time	119	241	242	245	271	294	319
Data Sorted (GB)	3.87	7.74	38.7	77.4	154.8	309.6	926.4
GB/s	0.03	0.03	0.16	0.32	0.57	1.16	2.90
	Local	One switch			More than one switch		

SkyServer

- Comparing the location and colour of stars in an astronomical table in Dryad and DryadLINQ
  - Dryad: 1000 lines of code in C++
  - DryadLINQ: 100 lines of code in C#
  - $1 < n < 40$

SkyServer


Dryad Two-pass
DryadLINQ
25.00
20.00
15.00
10.00
5.00
0.00
Speed-up
0
5
10
15
20
25
30
35
40
45
Number of computers

PageRank

- Simple PageRank (iterative hyperlinks counting)
  - Naïve: Links are grouped by source (one Join operation per page)
    - 93 lines of code
    - Scales well
    - 10 iterations in 12,792 seconds
  - Optimized: one Join operation per link (80-90% more local updates)
    - Scales well
    - 10 iterations in 690 seconds

Machine Learning

- Clustering algorithm
  - Parse and re-partition data across the cluster
  - Count the records
  - 10 iterations of E-M algorithm
  - Execution time: 7:11 minutes (5 hours of CPU processing)
- Statistical Inference Algorithm
  - Discover network-wide relationships between hosts and services
  - 4:22 hours (10 days of CPU processing)

DryadLINQ (+)

- Combining LINQ + Dryad
- User defined dataflow
- Stage fault tolerance
- Programming with C#/VB/F#
- Illusions of sequential application development
- Microsoft Visual Studio
- Support for other local LINQ execution engines
- Support for multiple storage systems (NTFS, SQL, Windows Azure, Cosmos DFS)
- .NET libraries

DryadLINQ (-)

- Create the illusion of developing for a single machine
- Dataflow cannot change after initializing
  - Vertices not able to spawn new vertices
- No support for data streaming and pipelining
  - Not suitable for real-time applications
- No support for debugging on the cluster
  - Only local simulation
- Evaluation could be better

Future Work

- Approach the main goal as much as possible:
  - Create the illusion of developing for a single machine
- Developing extensions for DryadLINQ
- Debugging on the cluster and performance debugging
- Reusing previous computed results
  - DryadInc: Reusing work in large-scale computations (2009)