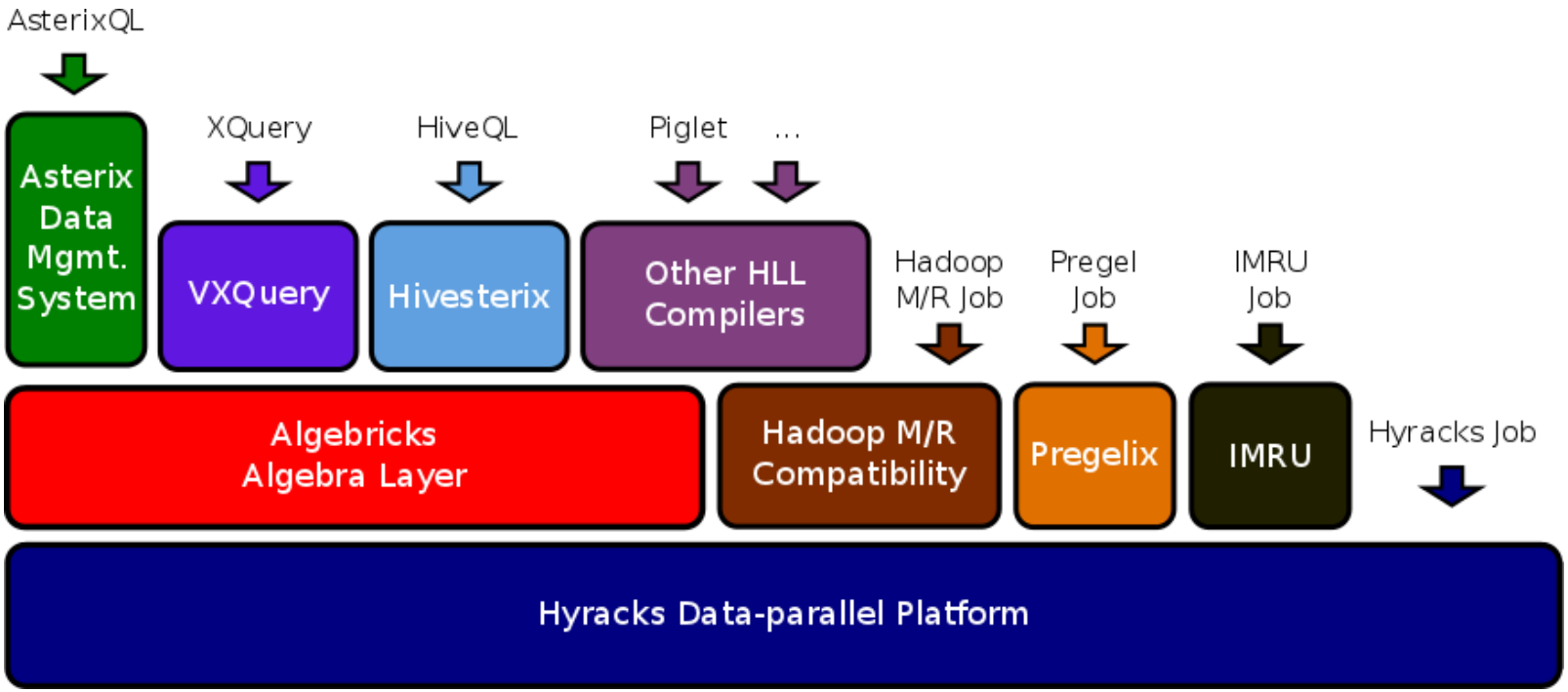
Making Sense of System Performance at Scale

Vinayak Borkar

UC Irvine

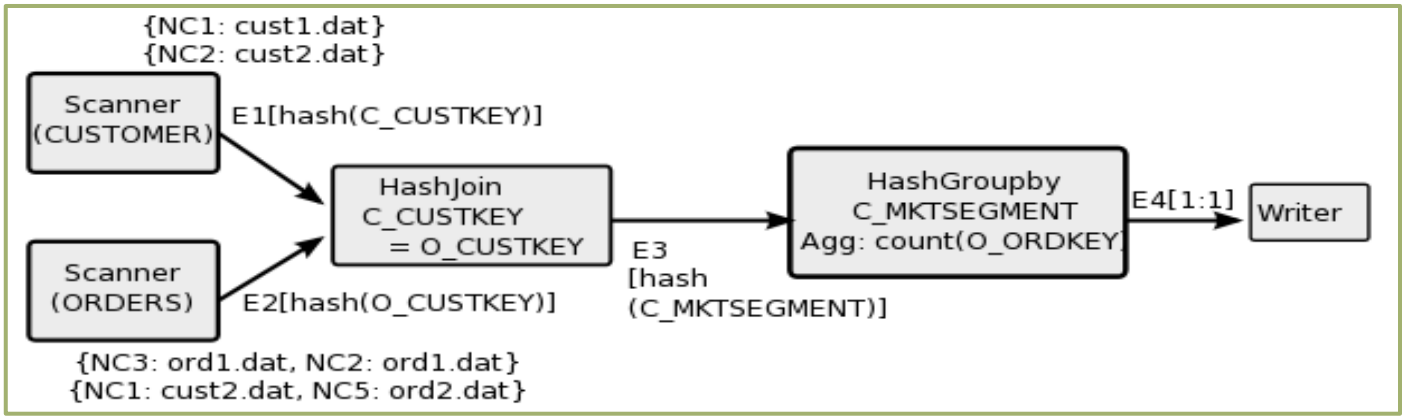
Joint work with Yingyi Bu

The ASTERIX Project

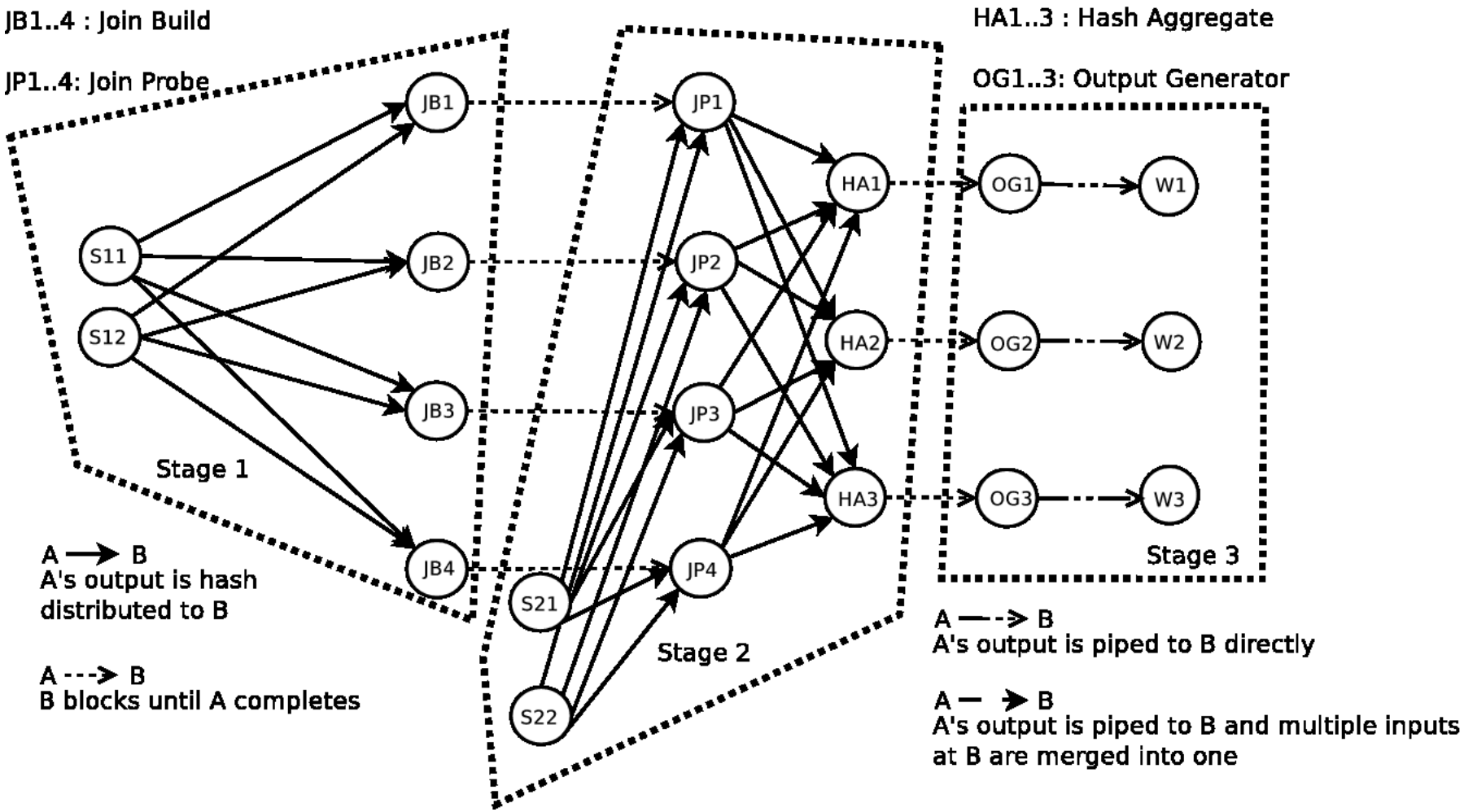


Hyracks In a Nutshell

- Partitioned-parallel platform for data-intensive computing
- Job = dataflow DAG of operators and connectors
  - Operators consume/produce partitions of data
  - Connectors repartition/route data between operators



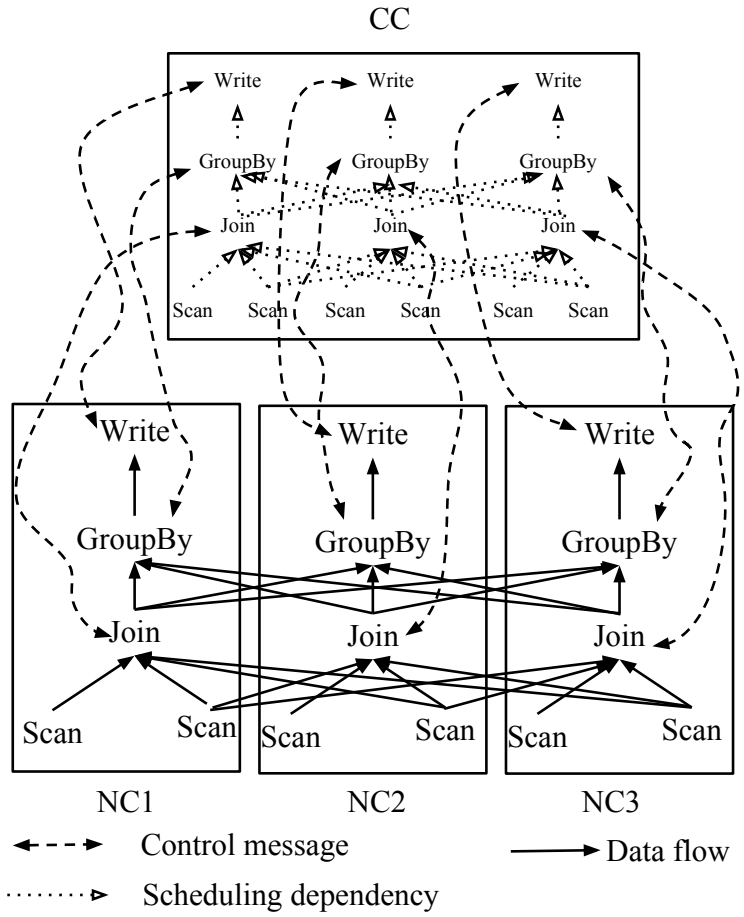
Hyracks: Runtime Task Graph



Hyracks Library (growing…)

- Operators
  - File readers/writers: line files, delimited files, HDFS files
  - Mappers: native mapper, Hadoopmapper
  - Sorters: in-memory, external
  - Joiners: in-memory hash, hybrid hash, grace hash
  - Aggregators: hash-based, preclustered
  - BTree Bulk Load, Search, Scan
- Connectors
  - M:N hash-partitioner
  - M:N hash-partitioning merger
  - M:N range-partitioner
  - M:N range-partitioning merger
  - M:N replicator
  - 1:1

Hyracks System Architecture



Hyracks (from 2009 to 2011)

- Built at UCI from the ground up
- Tested on the infrastructure we had:
  - 10 machines, 4 cores each, 12GB / machine
  - Single Rack with 1 GigE network
  - 4 spinning disks on each machine

Hyracks on the Yahoo! Cluster

- 180 machines, 8 cores each, 16GB / machine
- 6 racks, 1GigE between machines, 1GigE top-of-rack
- 4 Spinning disks per machine

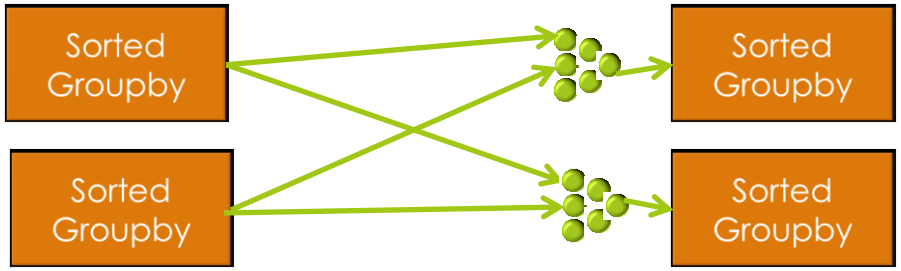
- The largest job was 1440-way parallel (as many cores)
- 1440 * 1440 = 2,073,600 logical data flows

Hyracks – Initial implementation

- Use Java RMI for control messages
  - Easy to implement
  - No surprise that it was a time bomb waiting to explode
- Use a TCP connection for each logical data flow
  - Gives excellent isolation across different flows
  - Hadoop has the same design, but no pipelining

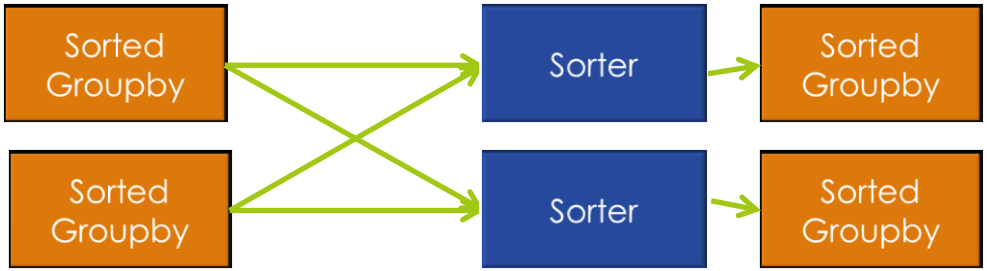
Example: Parallel Aggregation

Hash Partitioning
with Sorted Merging

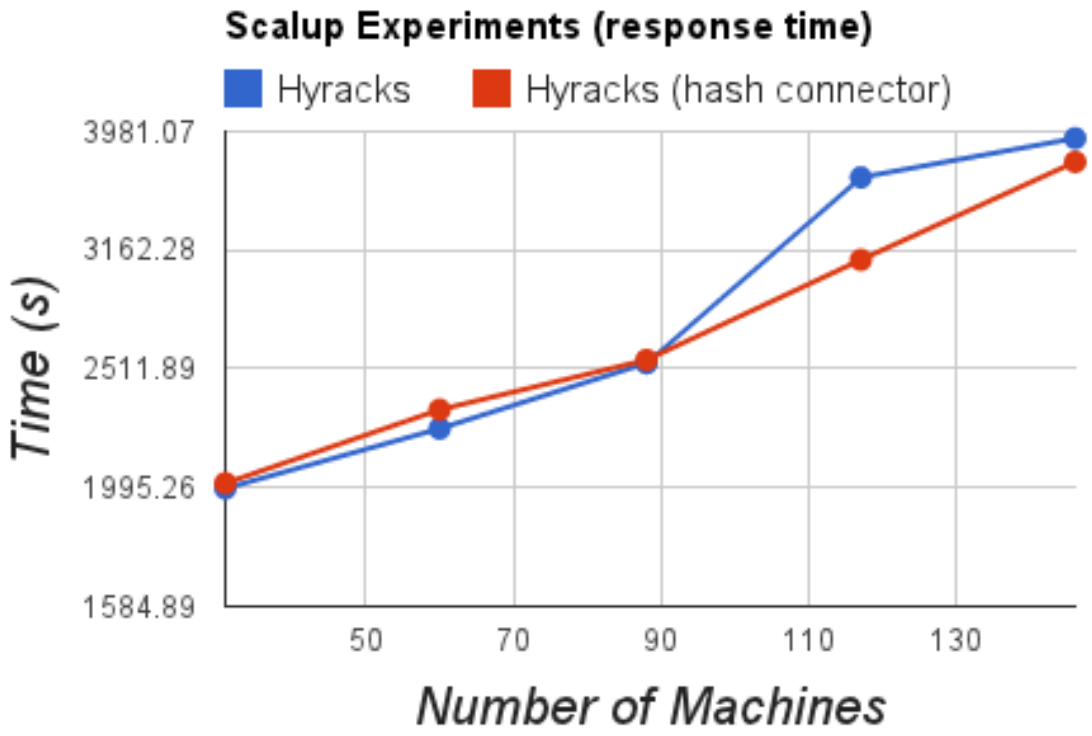


OR

Hash Partitioning
with Random Merging



Parallel Aggregation



Analyzing System Behavior

- Building efficient scalable systems is hard
- Need a way to understand system behavior at scale
- Tools for debugging/profiling distributed systems are MIA
- Each machine has a “local” view of events on the cluster
- Need to piece together “local views” to understand system behavior

Our direction

- Look at local performance information/logs as a partitioned database
- Use the capabilities of the Big Data Management System to analyze system performance
- Challenge: How do you collect all the information without creating the "observer effect"?